PROGRAMMING MANUAL

Version 1.12

DMG-06-4216-001-D
Released: 10/10/2000

“Confidential”

This document contains confidential and proprietary information of Nintendo and is also protected under the copyright laws of the United States and foreign countries. No part of this document may be released, distributed, transmitted or reproduced in any form or by any electronic or mechanical means, including information storage and retrieval systems, without permission in writing from Nintendo.

© 1999, 2000 Nintendo of America Inc.

TM and ® are trademarks of Nintendo

INTRODUCTION

This manual is a combination and reorganization of the information presented in the Game Boy Development Manual, revision G, and the Game Boy Color User's Guide, version 1.3. In addition, it incorporates all information related to Game Boy programming, including programming for Super Game Boy and the Game Boy Pocket Printer.

The abbreviations used in this manual represent the following:

DMG: Game Boy (monochrome), introduced on April 21, 1989
MGB: Game Boy Pocket (monochrome), introduced on July 21, 1996
MGL: Game Boy Light (monochrome), introduced on April 14, 1998
CGB: Game Boy Color (color), introduced on October 21, 1998

Note:	***Where it is not necessary to distinguish between the different monochrome models, DMG is used to refer to both monochrome models, and CGB is used to denote the Game Boy Color. Only where it is necessary to distinguish between the monochrome models is MGB used to denote Game Boy and MGL used to denote Game Boy Light.***

SGB: Super Game Boy, introduced on June 14, 1994
SGB2: Super Game Boy 2, introduced on January 30, 1998

Note:	***SGB is used to denote both SGB and SGB2 when no distinction is necessary. SGB2 is used only in cases where distinction is necessary.***

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

PREFACE: TO PUBLISHERS

Nintendo Game Boy Color Software Pre-Approval Requirements

Prior to submitting your CGB software to Lot Check for approval, it is required that you submit it to the Licensee Product Support Group for pre-approval. To assist us with the evaluation of your CGB software and/or product proposal(s), please refer to the following requirements when submitting materials* for approval.

* Please do not send original artwork or materials, as they will not be returned.

CGB software and/or product proposals are evaluated based on the following criteria:

- Use of Color
  To ensure that the expectations of the Game Boy Color consumer are met, Mario Club will evaluate the use of color in all CGB games (dual or dedicated) using the following criteria:

  ◊ **Differentiation -** If a game is to be considered CGB-compatible, then it must appear significantly more colorful than a monochrome Game Boy game when "colorized" by the CGB hardware. The principal measure of this is the number of colors in the background (BG) and the number of colors in the objects (OBJ).
  ◊ **Simultaneous Colors -** Because CGB hardware automatically "colorizes" monochrome games with up to four colors in the BG palette and up to six colors for two OBJ palettes (three colors per palette), a game typically must display more colors than this automatic "colorization" to be considered a CGB game.
  ◊ **Appropriate use of Color -** Objects in the game that are based on reality (trees, rocks, animals, and so on) should be a color that we would normally associate with them. For fictional objects, colors should be chosen to show appropriate detail and, when needed, to differentiate unlike objects.
  ◊ **Variety of Colors -** The CGB is capable of producing a wide range of colors (32,768 to be exact -- albeit not all at the same time). A CGB game should use this capability of the hardware to yield distinctly different colors for objects, characters, areas, and so on.
  ◊ **Contrast & Saturation -** Two of the elements that make a game look colorful are high contrast and "saturated" or vibrant colors. Pastel colors on a white background will not seem nearly as colorful as the same colors on a dark background. Not every game can use a dark background, but the intensity of the colors should still be maximized as much as possible.

  Please detail or demonstrate how your game will utilize color capabilities of the CGB. Use whatever means will best allow you to do so, such as artists renderings, programmed demos, ROM images, written descriptions, and so on.

- Game Concept content
  We do not require an explanation of, or evaluate game concept content for original CGB titles. However, if you are planning to "colorize" a previously released monochrome game we require that it include game-play enhancements (beyond simply adding color) to differentiate it from its monochrome counterpart. Such game-play enhancements may include, but are not limited to: additional stages, levels, or areas; new characters; additional items; game-play based on color; and so on. These enhancements must be readily apparent to players familiar with the original monochrome game.

  Please submit a written proposal of the enhancements to us for pre-approval. Use whatever additional means that will best allow you to communicate the game-play enhancements, such as storyboards, treatments, videotapes, programmed demos, and so on.

- Interim ROM Submissions
  We require at least one interim ROM submission to Mario Club (at approximately 50% completion) for preliminary review of the use of color in every CGB game. By reviewing the interim ROM and providing you with feedback in the early stages, we also help ensure that your projects stay on schedule. Final pre-approval is based on Mario Club's evaluation of a ROM near completion of game development.

  If you wish to arrange electronic transfer of the ROM image, please contact Terral Dunn in our Testing and Engineering department at (425) 861-2670 or by e-mail at "Terraldu@noa.nintendo.com". Please notify him when you have made an electronic submission for our review.

- Proposed Developer
  Please supply us with the name, address and phone number of the proposed developer. If the developer is not an Authorized Nintendo CGB Developer, please contact Melody Morgan at "melomo01@noa.nintendo.com" or 425-861-2618, and he will provide you with the application information.

- Schedule Information
  Please provide us with an estimated product schedule, including interim ROM submission(s), final Mario Club submission, submission of the master ROM to Lot Check, and the release date.

- Game Pak Configuration & Game Type
  Please provide us with the estimated Game Pak size in Megabits (Mb) and the RAM size if internal memory is to be used to save game information. Also state whether the game will be compatible with the monochrome Game Boy hardware or if it is dedicated to CGB hardware. For the current Game Pak prices and configurations available, please contact Nintendo's Licensing Department.

You will be contacted with the evaluation results when the Licensee Product Support Group has completed its evaluation of your ROM or concept submission.

Table of Contents

Introduction 3
Preface: To Publishers.....5
Chapter 1 System 11
Chapter 2 Display Functions.....49
Chapter 3 Sound Functions 81
Chapter 4 CPU Instruction Set.....97
Chapter 5 Miscellaneous General Information.....131
Chapter 6 The Super Game Boy System 143
Chapter 7 Super Game Boy Sound 203
Chapter 8 Game Boy Memory Controllers(MBC) 233
Chapter 9 Pocket Printer 255
Appendix 1 Programming Cautions.....273
Appendix 2 Register and Instruction Set Summaries.....289
Appendix 3 Software Submission Requirements 307

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 1: SYSTEM..... 11

1.1 General System 11

1.1.1 System Overview..... 11
1.1.2 Game Boy Block Diagram 13
1.1.3 Memory Configuration 14
1.1.4 Memory Map.....15
1.1.5 Feature Comparison.....16
1.1.6 Comparison of Registers 17

1.2 CPU..... 18

1.2.1 Overview of CPU Features.....18
1.2.2 CPU Block Diagram.....20
1.2.3 Description of CPU Functions.....21
1.2.4 CPU Functions (Common to DMG/CGB①).....24
1.2.5 CPU Functions (Common to DMG/CBG②).....29
1.2.6 CPU Functions (CGB only).....35

Revision History

Date	Section	Description
12/3/99	1.2.6.3	Revision of description for Infrared Communication

CHAPTER 1: SYSTEM

1.1 General System information

1.1.1 System Overview

Structure

At the heart of the DMG/CGB system is a CPU with a built-in LCD controller designed for DMG/CGB use.

System

[DMG]	[CGB]
➢ Dot-matrix LCD unit capable of grayscale display ➢ 64 Kbit – SRAM (for LCD display) ➢ 64 Kbit – SRAM (working memory)	➢ Color dot-matrix LCD unit capable of RGB with 32 grayscale shades ➢ 128 Kbit – SRAM (for LCD display) ➢ 256 Kbit – SRAM (working memory) ➢ Infrared communication link (photo transistor, photo LED)

Features common to DMG/CGB

- 32-pin connector (for ROM cartridge connection)
- 6-pin subconnector (for external serial communication)
- DC-DC converter for power source
- Sound amp
- Keys for operation
- Speaker
- Stereo headphone connector
- Input connector for external power source

Types of Game Pak Supported

1 Game Boy Game Pak

(Software that uses only the Game Boy functions. When used with Game Boy Color, 4-10 colors are displayed.)

2 Game Boy Color Game Pak

- Game Pak supported by CGB (for use with both CGB and DMG)
- Game Pak for CGB only (software that runs only on CGB)

Operating Modes (the following modes apply only to CGB)

1 DMG Mode (when using software for DMG)
The new registers, expanded memory area, and new features for CGB are not used. Color applications previously associated with palette data BGP, OBP0, and OBP1 are performed by the system.

2 CGB Mode (when using software supported or used exclusively by CGB)
The new registers, expanded memory area, and new features of CGB are available.

Note:	***To operate in CGB mode, specific code must first be placed in the ROM data area of the user program. For more information, see Chapter 5, Section 2,* Recognition of CGB support (CGB only) in ROM Data.**

Power Source

- Battery/AC adapter/Battery charger

Accessories (as of April 1999)

DMG Accessories

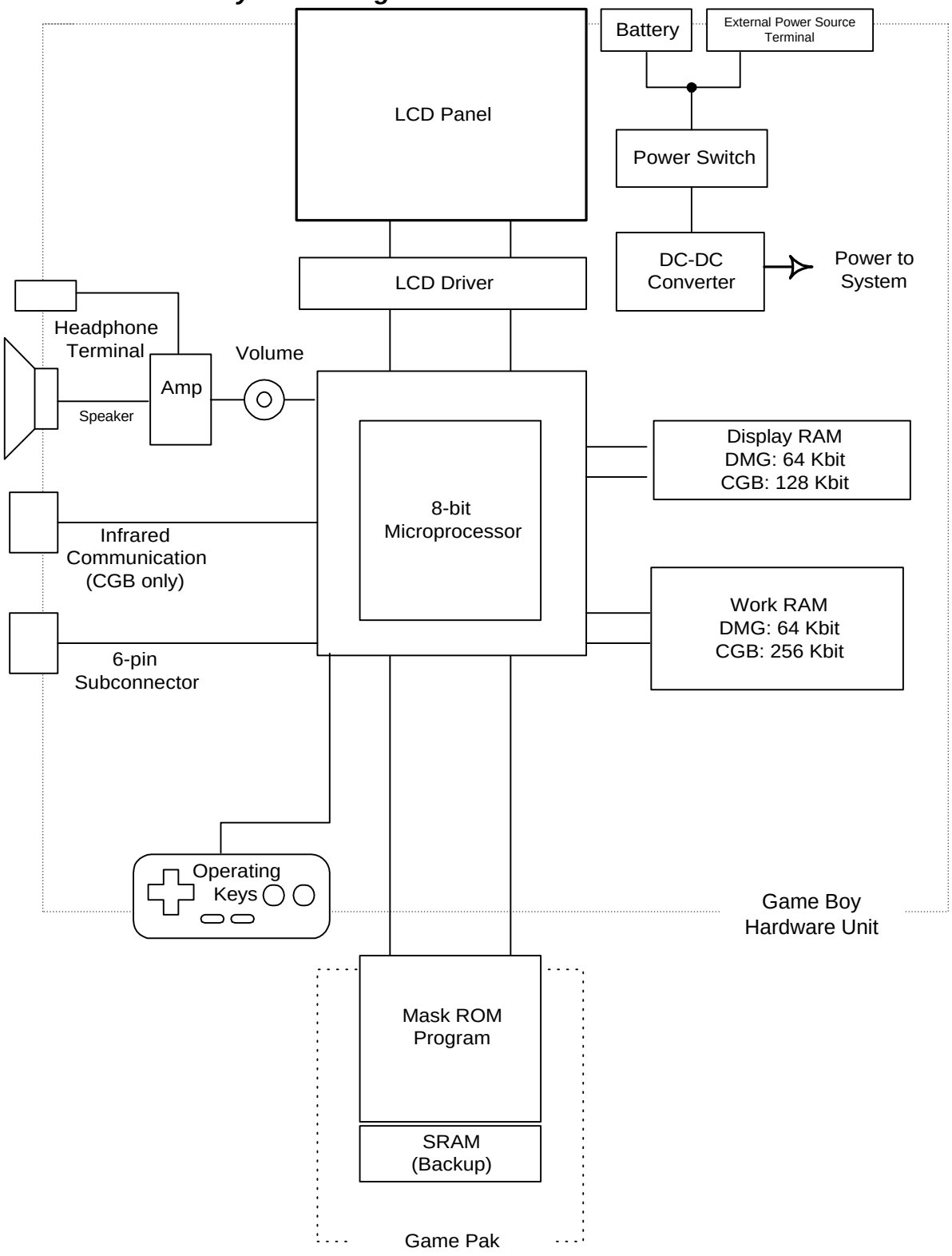
- Communication Cable
- Battery Charger Adapter

MGB/CGB Accessories

- Communication Cable
- AC Adapter
- Battery Pack Charger Set

The 6-pin serial communication subconnector and the AC adapter input connector of the DMG hardware that preceded MGB are shaped differently than those of MGB and CGB. Thus, two types of accessories are available — those exclusively for DMG and those exclusively for MGB/CGB. In addition, a conversion connector is necessary for communication between DMG and MGB/CGB.

1.1.2 Game Boy Block Diagram



1.1.3 Memory Configuration

In DMG and CGB, the 32 KB from 0h to 7FFFh is available as program area.

000h-0FFh: Allocated as the destination address for RST instructions and the starting address for interrupts.
100h-14Fh: Allocated as the ROM area for storing data such as the name of the game.
150h: Allocated as the starting address of the user program.

The 8 KB from 0x8000 to 0x9FFF is used as RAM for the LCD display. In CGB, the amount of RAM allocated for this purpose is 16 KB (8 KB x 2), twice the amount allocated for the LCD display in DMG, and this RAM can be used in 8 KB units using bank switching. The 8 KB RAM areas are divided into the following 2 areas.

1 An area for character data
2 An area for BG (background) display data (Character code and attribute)

The 8 KB from A000h to BFFFh is the area allocated for external expansion RAM.
The 8 KB from 0xC000 to 0xDFFF is the work RAM area.
In DMG, the 8 KB of working RAM is implemented without change. In CGB, bank switching is used to provide 32 KB of working RAM. This 32 KB area is divided into 8 areas of 4 KB each.

1 The 4 KB from C000h to CFFFh is fixed as Bank 0.
2 The 4 KB from D000h to DFFFh can be switched between banks 1 though 7.

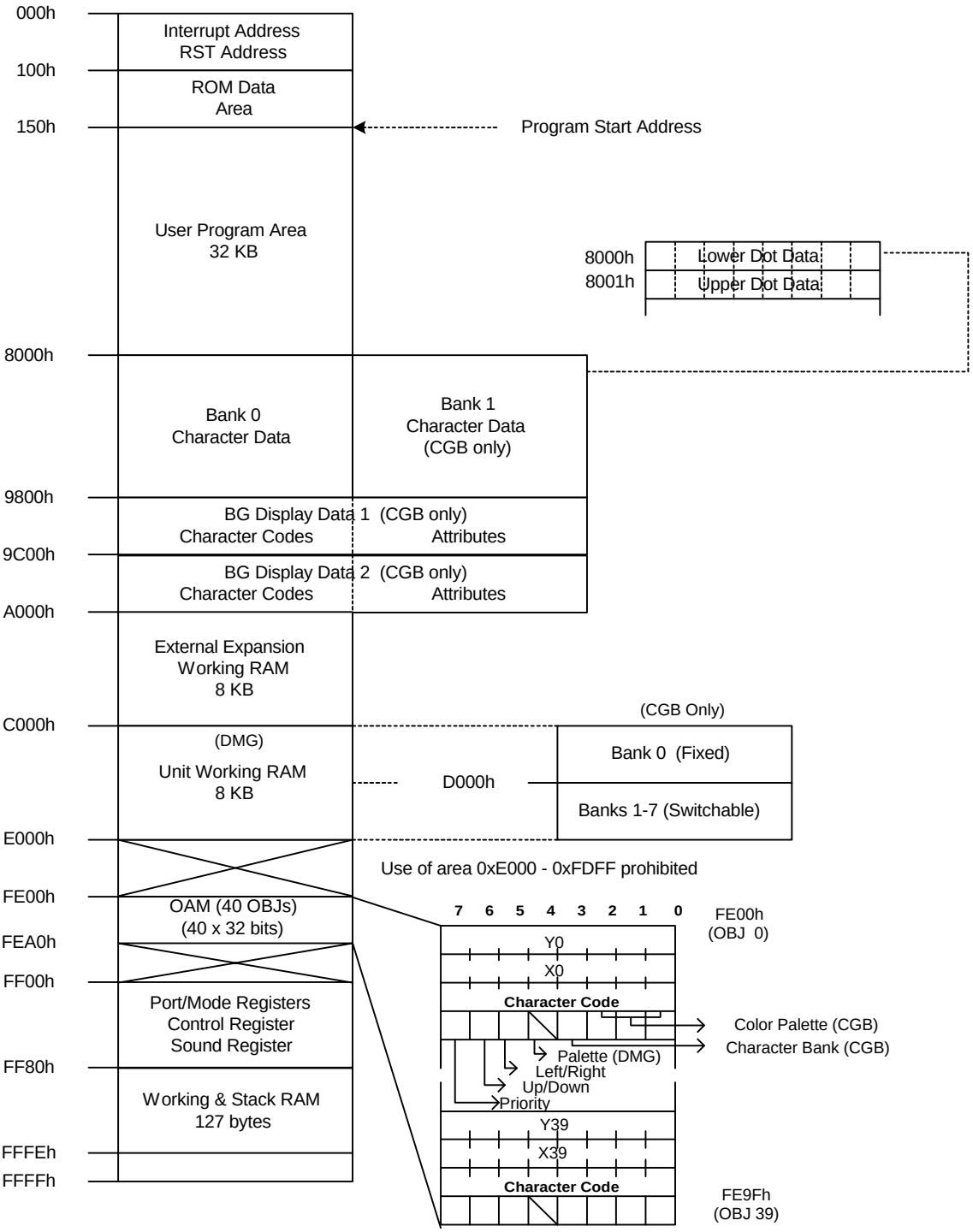
Note: Use of the area from E000h to FDFFh is prohibited.

FE00h to FFFFh is allocated for CPU internal RAM.

FE00h-FE9Fh: OAM-RAM (Holds display data for 40 objects)
FF00h-FF7Fh & FFFFh: Specified for purposes such as instruction registers and system controller flags.
FF80h-FFFEh: Can be used as CPU work RAM and/or stack RAM.

1.1.4 Memory Map

Note: In DMG, there is no bank switching at 8000h-9FFFh and C000h-DFFFh.



1.1.5 Feature Comparison

Item	DMG CPU	CGB CPU
CPU Speed (system operating frequency)	1.05 MHz	1.05 MHz (normal mode) 2.10 MHz (double-speed mode)
Game Boy RAM		
Work and Stack RAM	127 x 8 bits	←
Work RAM	8,192 bytes	32,768 bytes
OAM	40 x 28 bits	40 x 32 bits
For LCD display	8,192 bytes	16,384 bytes
Game Pak Memory Space		
ROM (without MBC)	32,768 bytes	←
RAM (without MBC)	8,192 bytes	←
LCD Controller		
Display Capacity	160 x 144 dots	160 x 144 x RGB dots
Block Structure		
BG, window	8 x 8 dots	←
Object	8 x 8 dots or 8 x 16 dots	←
Number of Usable Characters		
BG	256	512
OBJ 8 x 8	256	512
8 x 16	128	256
Grayscale: BG, window	4 shades, 1 palette	4 colors, 8 palettes (DMG mode: 4 colors, 1 palette)
Grayscale: Object	3 shades, 2 palettes	3 colors, 8 palettes (DMG mode: 3 colors, 2 palettes)
Object priority		
Different x coordinates	Object with smallest x coord .	Object with lowest OBJ number (DMG mode: Object with lowest x coord.)
Same x coordinates	Object with lowest OBJ number	←
Timer & Divider Stages	8-bit timer x 1 16 stages x 1	← ←
Serial Input/Output	8 bits x 1	←
Baud Rate	8 K	8K/256K (16K/512K in high-speed mode
DMA Controller		
Existing DMA	8000h~DFFFh→OAM	0h~DFFFh→OAM
Horizontal blank DMA	---	Game Pak & Work RAM→VRAM
General-purpose DMA	---	Game Pak & Work RAM→VRAM
Interrupt features		
Internal Interrupts	4 types (maskable)	←
External Interrupts	1 type (maskable)	←
Input/Output Ports		
Serial Input/Output Ports	SIN, SCK, SOUT	←
Infrared Communication Port	---	R0, R1, R2, R3
Sound Output Circuit	4 sounds	← Monaural (VIN) External Sound Mixable Input

←: Same as in column at left

1.1.6 Comparison of Registers

	DMG CPU		CGB CPU	
Use	**Register**	**Address**	**Register**	**Address**
Port/Mode	P1	FF00	←	←
Registers	SB	FF01	←	←
	SC	FF02	←	←
	DIV	FF04	←	←
	TIMA	FF05	←	←
	TMA	FF06	←	←
	TAC	FF07	←	←
		---	KEY1	FF4D
		---	RP	FF56
Bank Control		---	VBK	FF4F
Registers		---	SVBK	FF70
Interrupt	IF	FF0F	←	←
Flags	IE	FFFF	←	←
	IME		←	
LCD Display	LCDC	FF40	←	←
Registers	STAT	FF41	←	←
	SCY	FF42	←	←
	SCX	FF43	←	←
	LY	FF44	←	←
	LYC	FF45	←	←
	DMA	FF46	←	←
	BGP	FF47	←	←
	OBP0	FF48	←	←
	OBP1	FF49	←	←
	WY	FF4A	←	←
	WX	FF4B	←	←
		---	HDMA1	FF51
		---	HDMA2	FF52
		---	HDMA3	FF53
		---	HDMA4	FF54
		---	HDMA5	FF55
		---	BCPS	FF68
		---	BCPD	FF69
		---	OCPS	FF6A
		---	OCPD	FF6B
	OAM	FE00~FE9F	←	←
Sound Registers	NR x x	FF10~FF26	←	←
	Waveform RAM	FF30~FF3F	←	←

←: Same as in column at left

1.2 CPU

1.2.1 Overview of CPU Features

The CPUs of DMG and CGB are ICs customized for DMG/CGB use, and have the following features.

CPU Features

Central to the 8-bit CPU are the following features, including an I/O port and timer.

- 127 x 8 bits of built-in RAM (working and stack)
- RAM for LCD Display: <DMG> 8 KB/<CGB>16 KB ( )
- Working RAM: <DMG> 8KB/<CGB> 32 KB
- Built-in 16-stage Frequency Divider
- Built-in 8-bit Timer
- 4 types of Internal Interrupts (maskable)
- 1 type of External Interrupt (maskable)
- Built-in DMA Controller
- Input Ports P10 ~ P13
- Output Ports P14 and P15
- Serial I/O Ports SIN, SCK, SOUT
- Infrared I/O Port <CGB only>

LCD Controller Functions

Game Boy is equipped with functions that provide control of the images displayed on the LCD. Character data used for display is held in system RAM.

- DMG: 4 shades of gray; CGB: 32 shades of gray for each RGB color
- 160 x 144-dot liquid crystal display
- 8 x 8-dot composition of background and window characters
- 8 x 8 or 8 x 16-dot composition of OBJ characters
- Up to 40 objects displayable in 1 screen
- Up to 10 objects displayable on 1 horizontal line
- 40 x 32 bits of built-in RAM (OBJ-RAM for LCD)
- Control of 256 x 256-dot background
- Vertically and horizontally scrollable background
- Window-like functions

Sound Functions

Each system is equipped with 4 types of sound synthesis circuitry.

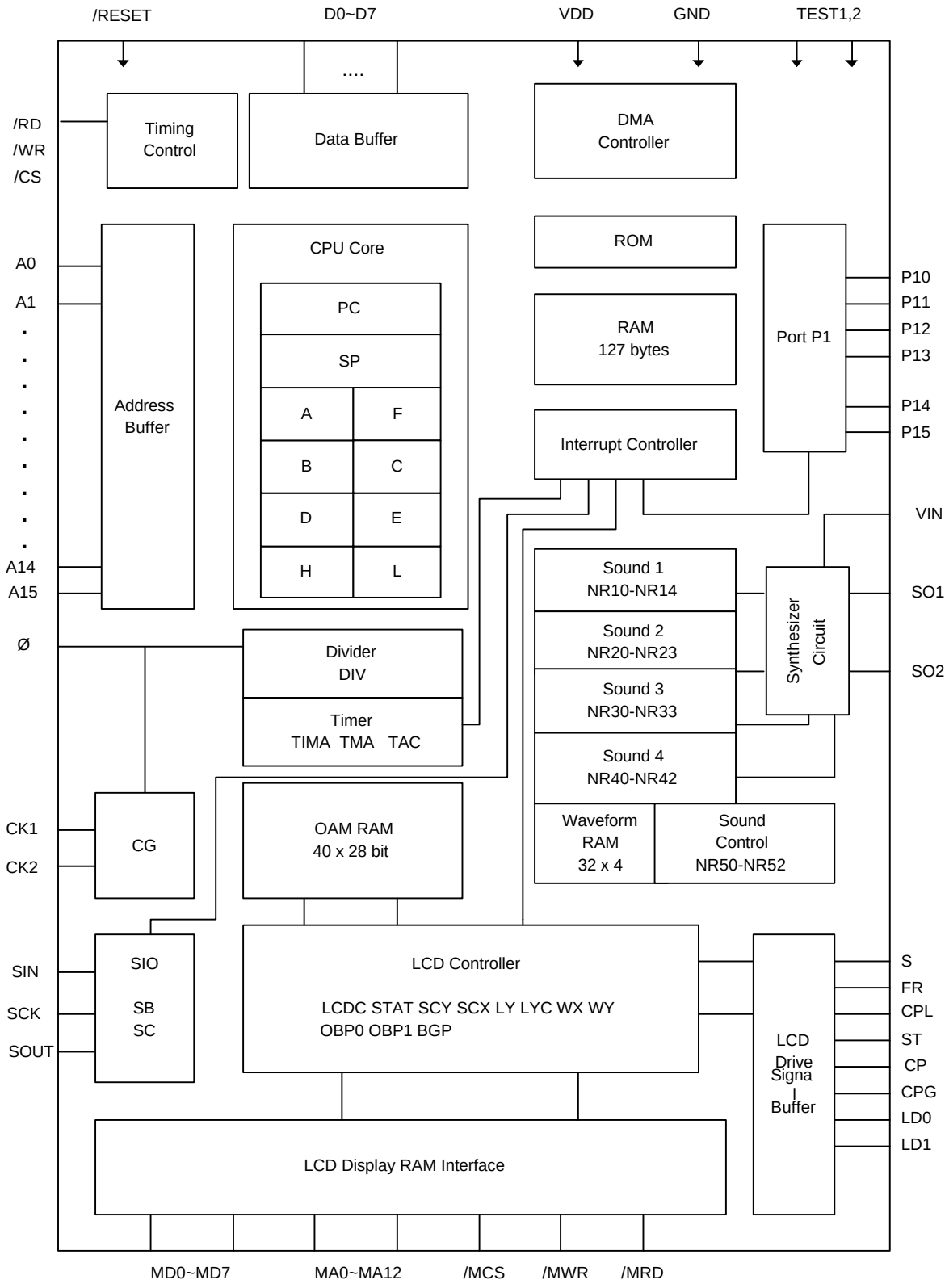
- Sound 1: Quadrangular waveform, sweep and envelope functions
- Sound 2: Quadrangular waveform, envelope functions
- Sound 3: Arbitrary waveform, generated
- Sound 4: White noise, generated
- 2 output channels (output can be allocated to a channel)
- Synthesized output with external sound input <CGB only>

Miscellaneous

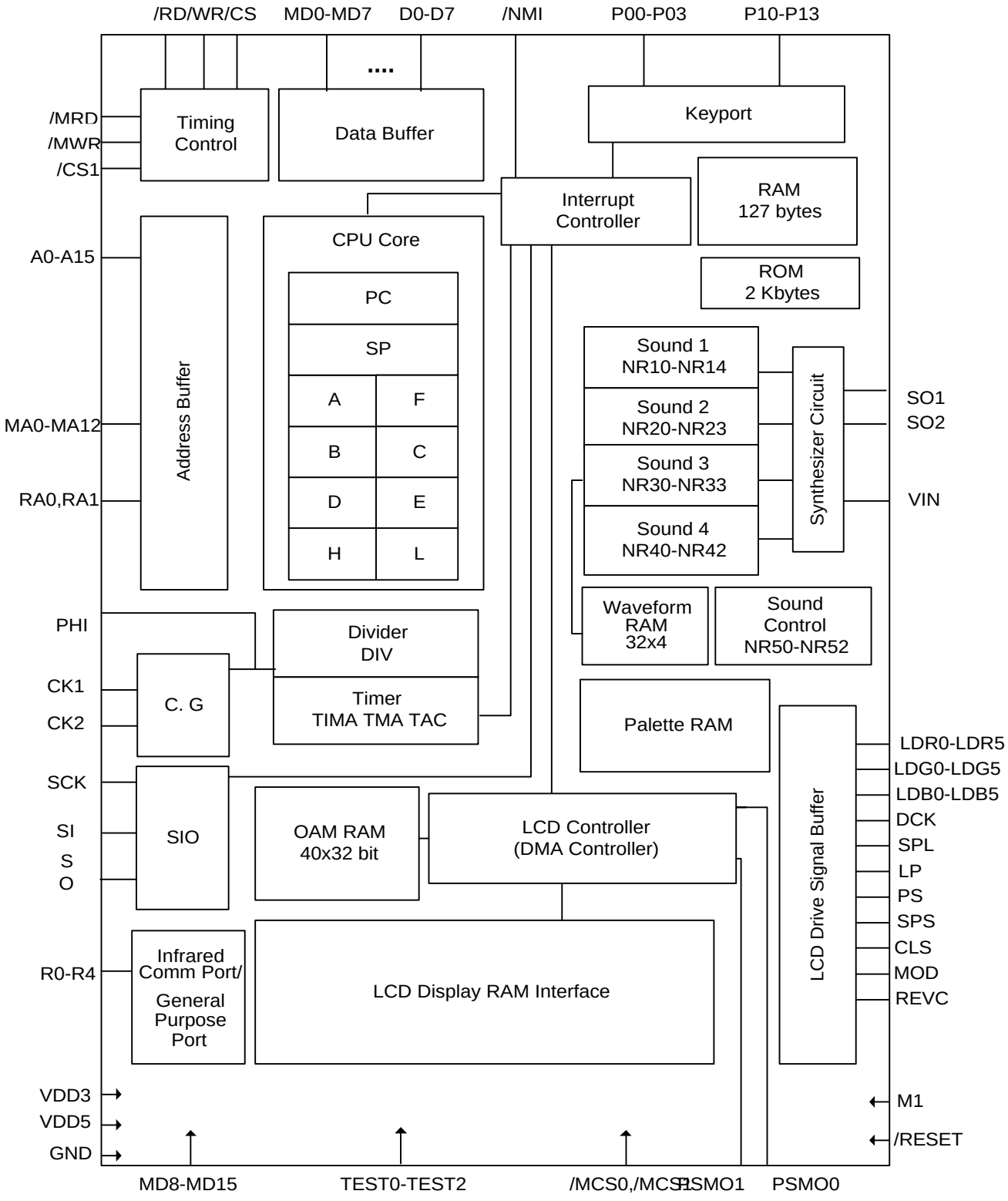
- An internal monitor program is built into DMG/CGB CPUs. When power is turned on or the Game Boy is reset, the internal monitor program first initializes components such as the ports, then passes control to the user program.
- Instruction cycles
  <DMG> 0.954 μs (source oscillation: 4.1943 MHz)
  <CGB> 0.954 μs/0.477 μs, switchable (source oscillation: 8.3886 MHz)

1.2.2 CPU Block Diagram

Game Boy (DMG/MGB) CPU



Game Boy Color CPU



1.2.3 Description of CPU Functions

Interrupts

There are five types of interrupts available, including 4 types of maskable internal interrupts and 1 type of maskable external interrupt. The IE flag is used to control interrupts. The IF flag indicates which type of interrupt is set.

- LCD Display Vertical Blanking
- Status Interrupts from LCDC (4 modes)
- Timer Overflow Interrupt
- Serial Transfer Completion Interrupt
- End of Input Signal for ports P10-P13

DMA Transfers

DMA transfers are controlled by the DMA registers.

<DMG>
DMG allows 40 x 32-bit DMA transfers from 8000h-DFFFh to OAM (FE00h-FE9Fh). The transfer start address can be specified in increments of 100h for 8000h-DFFFh.
<CGB>
In addition to the DMA transfers method for DMG (from 0000h-DFFFh in CGB), CGB enables two new types of DMA transfer — horizontal blanking and general-purpose DMA transfers.

Note, however, that when performing a DMG-type DMA transfer on CGB, some consideration must be given to specifying the destination RAM area.
For more information, see the DMA Functions section in Chapter 2.

1 Horizontal Blanking DMA Transfer
Sixteen bytes of data are automatically transferred for each horizontal blanking period during a DMA transfer from the user program area (0000h-7FFFh) or external and hardware working RAM area (A000h-DFFFh) to the LCD display RAM area (8000h-9FFFh).

2 General-Purpose DMA Transfer
Between 16 and 2048 bytes of data (specified in 16-byte increments) are transferred from the user program area (0000h-7FFFh) or external and hardware working RAM area (A000h-DFFFh) to the LCD display RAM area (8000h-9FFFh), during the Vertical Blanking Period.

Timer

The timer is composed of the following:

- TIMA (timer counter)
- TMA (timer modulo register)
- TAC (timer control register)

Controller Connections

- P10-P13: Input ports
- P14-P15: The key matrix structure is composed of the output ports.

At user program startup, the status of the CPU port registers and mode registers are as follows.

Register	Status
P1	0
SC	0
TIMA	0
TAC	0
IE	0
LCDC	$83 BG/OBJ ON, LCDC OPERATION
SCY	0
SCX	0
LYC	0
WY	0
W	0
Interrupt Enable (IE)	DI

Stack: FFFEh

Standby Modes

The standby functions are HALT mode, which halts the system clock, and STOP mode, which halts oscillation (source oscillation).

HALT Mode

Game Boy switches to HALT mode when a HALT instruction is executed.
The system clock and CPU operation halt in this mode. However, operation of source oscillation circuitry between terminals CK1 and CK2 continues. Thus, the functions that do not require the system clock (e.g,, DIV, SIO, timer, LCD controller, and sound circuit) continue to operate in this mode.
HALT mode is canceled by the following events, which have the starting addresses indicated.

1) A LOW signal to the /RESET terminal
   Starting address: 0000h
2) The interrupt-enable flag and its corresponding interrupt request flag are set
   IME = 0 (Interrupt Master Enable flag disabled)
   Starting address: address following that of the HALT instruction
   IME = 1 (Interrupt Master Enable flag enabled)
   Starting address: each interrupt starting address

STOP Mode

Game Boy switches to STOP mode when a STOP instruction is executed.
The system clock and oscillation circuitry between the CK1 and CK2 terminals are halted in this mode. Thus, all operation is halted except that of the SI0 external clock. STOP mode is canceled by the following events, and started from the starting address.

3) A LOW signal to the /RESET terminal
   Starting address: 0000h
4) A LOW signal to terminal P10, P11, P12, or P13
   Starting address: address following that of STOP instruction

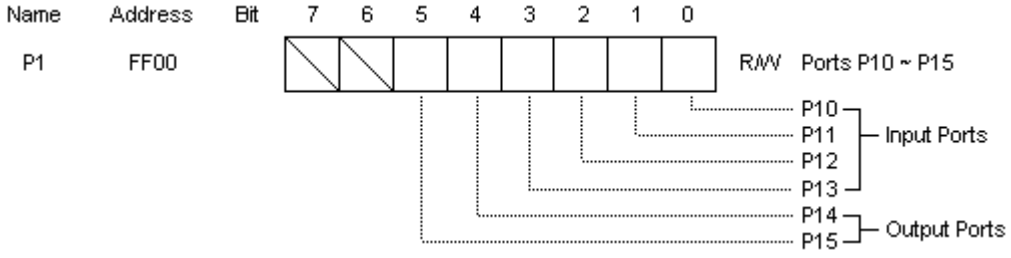
When STOP mode is canceled, the system clock is restored after 2^{17} times the oscillation clock (DMG: 4 MHz, CGB: 4 MHz/8 MHz), and the CPU resumes operation.
When STOP mode is entered, the STOP instruction should be executed after all interrupt-enable flags are reset, and meanwhile, terminals P10-P13 are all in a HIGH period.

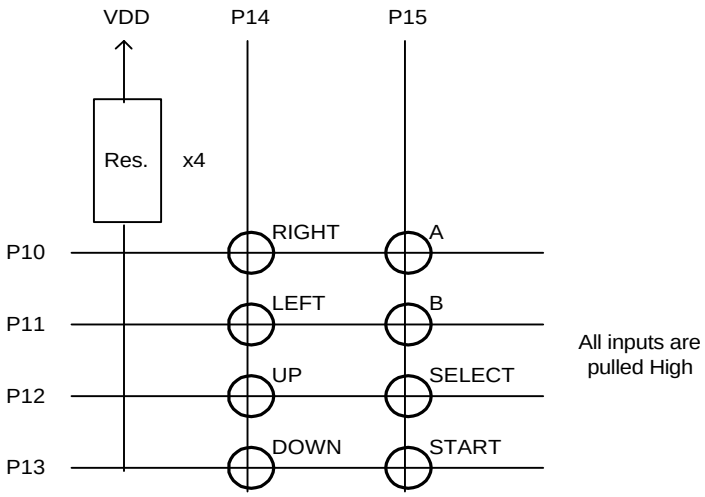
1.2.4 CPU Functions (Common to DMG/CGB①)

The CPU functions described here are those that are identical in DMG and CGB. CPU functions that are enhanced in CGB are described in Section 1.2.5, *CPU Functions (Common to DMG/CGB②)*. CPU functions that cannot be used for DMG are described in Section 1.2.6, *CPU Function (CGB only)*.

1.2.4.1 Controller Data

The P1 ports are connected with a matrix for reading key operations.





When key input is read, a brief interval is interposed between P14 and P15 output and reading of the input, as shown below.

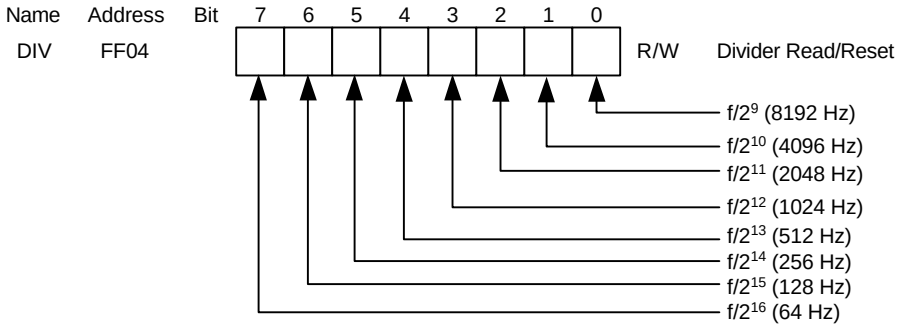
```
Example:  KEY     LD      A, $20          ; Read U, D, L, R keys
                  LD      ($FF00), A      ; Port P14  ← LOW output
                  LD      A, ($FF00)      ; A Register ← Port P10-P13
                  LD      A, ($FF00)      ; Perform this operation twice
                   .
                   .
                  LD      A, ($10)        ; Reads keys A, B, SE, ST
                  LD      ($FF00), A      ; Port P15 ← LOW output

                  LD      A, ($FF00)      ; A Register ← Ports P10-P13
                  LD      A, ($FF00)      ; Perform this operation 6 times
                  LD      A, ($FF00)      ;
                   .      ;
                   .      ;
                  LD      A, $30          ; Port reset
                  LD      ($FF00), A
                   .
                  RET
```

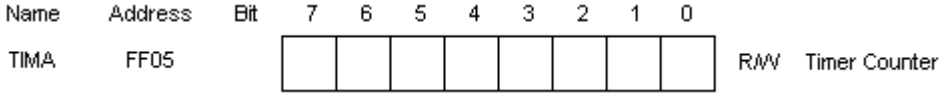
The interrupt request flag (IF: 4) is set by negative edge input at one of the P13-P10 terminals. Negative edge input requires a LOW period of 2^4 times source oscillation (DMG = 4 MHz, CGB = 4 MHz/8 MHz).

The interrupt request flag (IF: 4) also is set when a reset signal is input to the /RESET terminal with a P13~P10 terminal in the LOW state.

1.2.4.2 Divider Registers

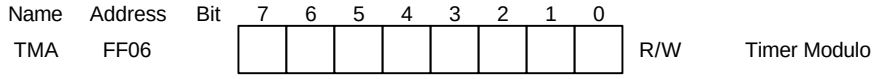


The upper 8 bits of the 16-bit counter that counts the basic clock frequency (f) can be referenced. If an LD instruction is executed, these bits are cleared to 0 regardless of the value being written (f = 4.194304 MHz).

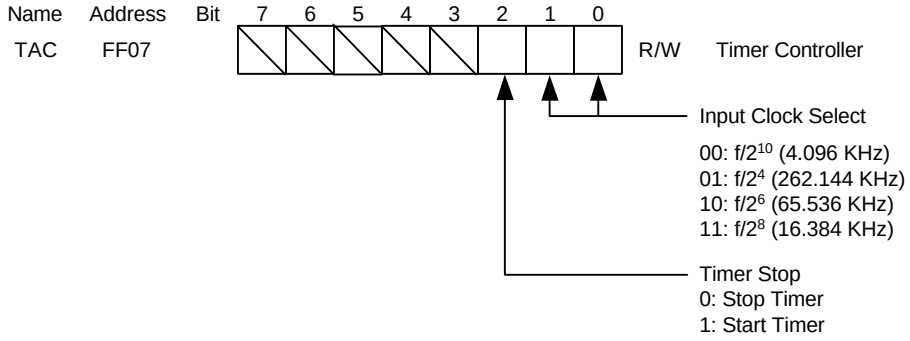


1.2.4.3 Timer Registers

The main timer unit. Generates an interrupt when it overflows.



The value of TMA is loaded when TIMA overflows.



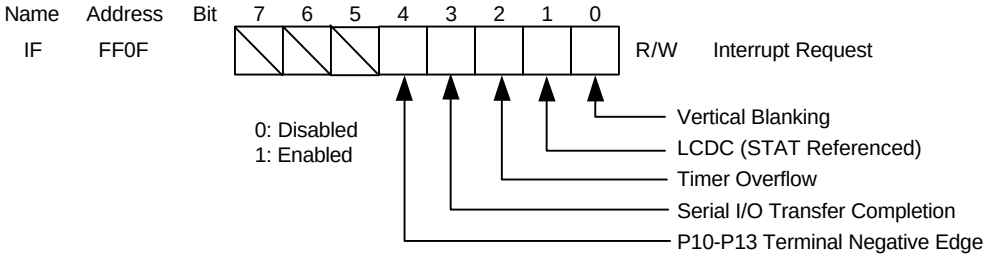
The timer consists of TIMA, TMA, and TAC.
The timer input clock is selected by TAC.
TIMA is the timer itself and operates using the clock selected by TAC.
TMA is the modulo register of TIMA. When TIMA overflows, the TMA data is loaded into TIMA.
Writing 1 to the 2nd bit of TAC starts the timer.
The timer should be started (the TAC start flag set) after the count up pulse is selected. Starting the timer before or at the same time as the count up pulse is selected may result in excessive count up operation.

Example:

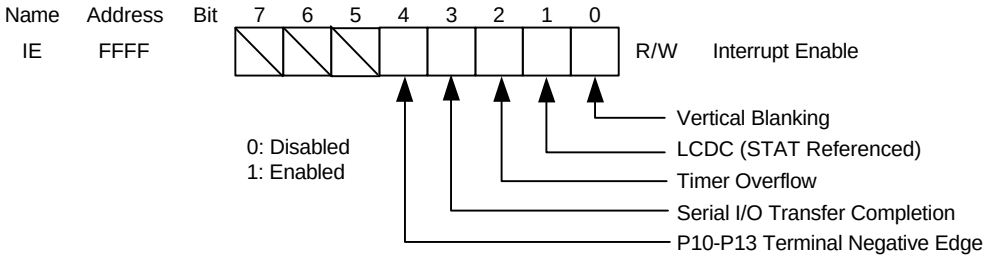
```
L D     A, 3          ;Select a count pulse of f/2^8
L D     (07), A       ;TAC ← 3 set
L D     A, 7          ;Start timer
L D     (07), A       ;
```

If a TMA write is executed with the same timing as that with which the contents of the modulo register TMA are transferred to TIMA as the result of a timer overflow, the same data is transferred to TIMA.

1.2.4.4 Interrupt Flags



Bit reset enabled



Name

IME Interrupt Master Enable
0: Reset by DI instruction, prohibits all interrupts
1: Set by EI instruction, the interrupts set by the IE registers are enabled

Bit reset enabled

Interrupts are controlled by the IE (interrupt enable) flag.
The IF (interrupt request) flag can be used to determine which interrupt was requested.

The 5 types of interrupts are as follows:

Cause of Interrupt	Priority	Interrupt starting address
Vertical blanking	1	0040h
LCDC status interrupt	2	0048h
Timer overflow	3	0050h
Serial transfer completion	4	0058h
P10-P13 input signal goes low	5	0060h

The LCDC interrupt mode can be selected (see STAT register).

Mode 00
Mode 01
Mode 10
LYC=LY
consist

When multiple interrupts occur simultaneously, the IE flag of each is set, but only that with the highest priority is started. Those with lower priorities are suspended.

When using an interrupt, set the IF register to 0 before setting the IE register.

The interrupt process is as follows:

1 When an interrupt is processed, the corresponding IF flag is set.
2 Interrupt enabled.
  If the IME flag (Interrupt Master Enable) and the corresponding IE flag are set, the interrupt is performed by the following steps.
3 The IME flag is reset, and all interrupts are prohibited.
4 The contents of the PC (program counter) are pushed onto the stack RAM.
5 Control jumps to the interrupt starting address of the interrupt.

The resetting of the IF register that initiates the interrupt is a hardware reset.

The interrupt processing routine should push the registers during interrupt processing.

When an interrupt begins, all other interrupts are prohibited, but processing of the highest level interrupt is enabled by controlling the IME and IE flags with instructions.

Return from the interrupt routine is performed by the RET1 and RET instructions.

If the RETI instruction is used for the return, the IME flag is automatically set even if a DI instruction is executed in the interrupt processing routine.

IF the RET instruction is used for the return, the IME flag remains reset unless an EI instruction is executed in the interrupt routine.

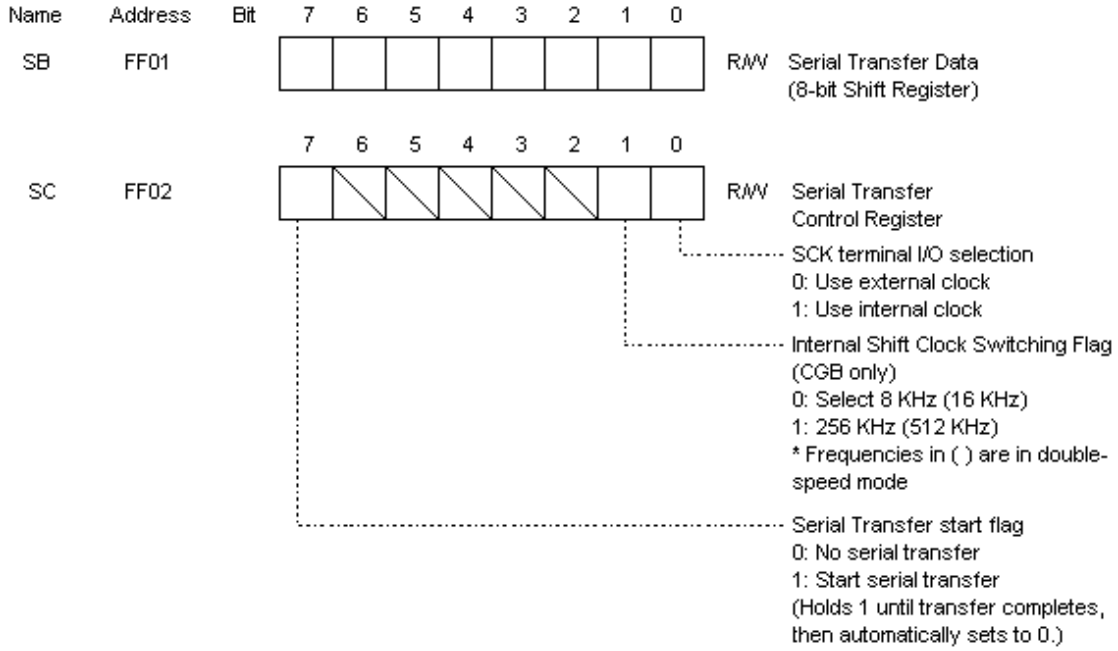
Each interrupt request flag of the IF register can be individually tested using instructions.

Interrupts are accepted during the op code fetch cycle of each instruction.

1.2.5 CPU Functions (Common to DMG/CGB②)

This section describes the CPU functions that have been enhanced in CGB. Functions that are identical in DMG and CGB are described in Section 1.2.4, *CPU Functions (Common to DMG/CGB①)*. CPU functions not available in DMG are described in Section 1.2.6, *CPU Functions (CGB only)*.

1.2.5.1 Serial Cable Communication



Note: ***In DMG mode, bit 1 of the SC register is set to 1 and cannot be changed, but the transfer speed is fixed at 8 KHz.***

Serial I/O (SIO) is controlled by the SB and SC registers.
The lowest bit (SC0) of the SC register can be used to select shift clock to be either the external clock from the SCK terminal or the internal shift clock.
Sending and receiving occur simultaneously with a serial transfer.
If the data to be sent is set in the SB register and the serial transfer is then started, the received data is set in the SB register when the transfer is finished.

Serial transfer procedure:

1. The data is set in the SB register.
2. Setting the highest SC register bit (SC 7) to 1 starts the transfer.
3. The 3-bit counter is reset and after 8 counts of the shift clock, the transfer is performed until overflow occurs.
4. SC7 is reset.
5. If the serial transfer completion interrupt is enabled, the CPU is interrupted.

When the shift clock goes low, the contents of the SB register are shifted leftward and the data is output from the highest bit. When the shift clock goes high, input data from the SIN terminal are output to the lowest bit of the SB register.

When the SCK terminal is in external-clock mode, it is pulled up to VDD.

If the highest bit of the SC register (SC7) is set, reading and writing to the SB register is prohibited.

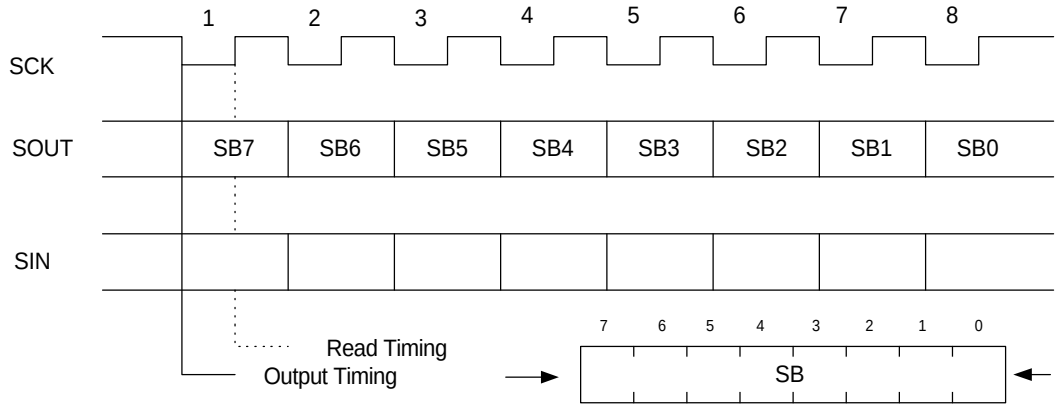
An SIO serial transfer should be started (highest SC bit set) after the external or internal shift clock is selected. Excessive shifting may result if the transfer is started before or at the same time as the shift clock is selected.

If a transfer is performed using the external clock, the data is first set in the SB register, then the SC register start flag is set and input from the external clock is awaited. The transfer start flag must be set each time data is transferred.

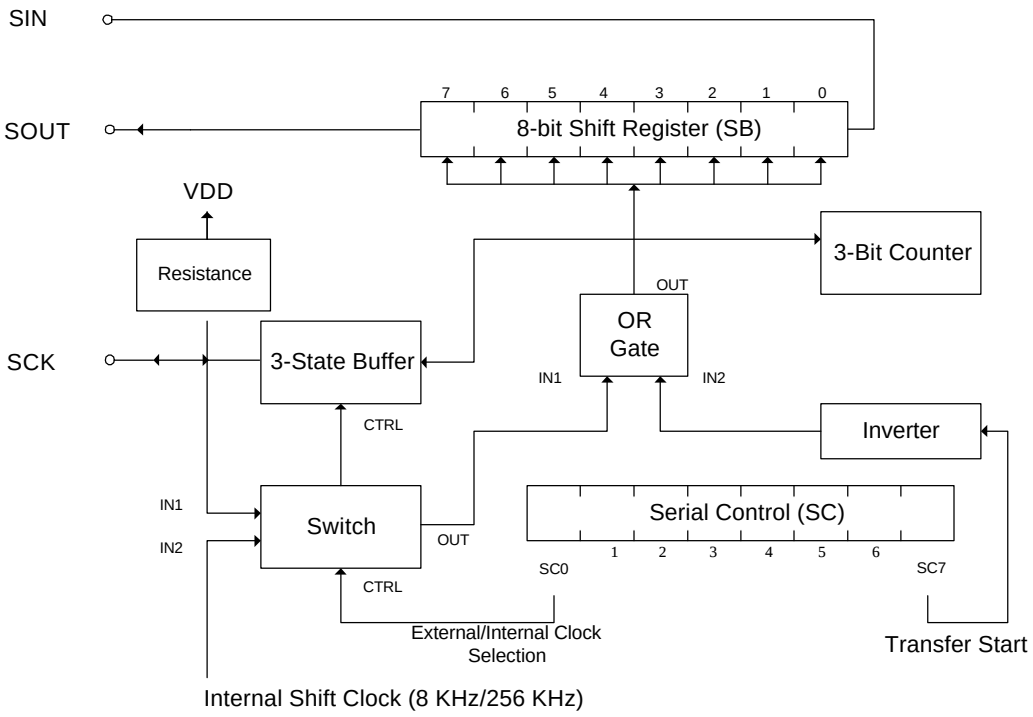
The maximum setting for an external clock is 512 KHz.

Serial communication (SIO) specifications are essentially the same for DMG and CGB. In CGB, however, the operating speed of the internal shift clock can be set to high by specifying a speed in bit 1.

SIO Timing Chart

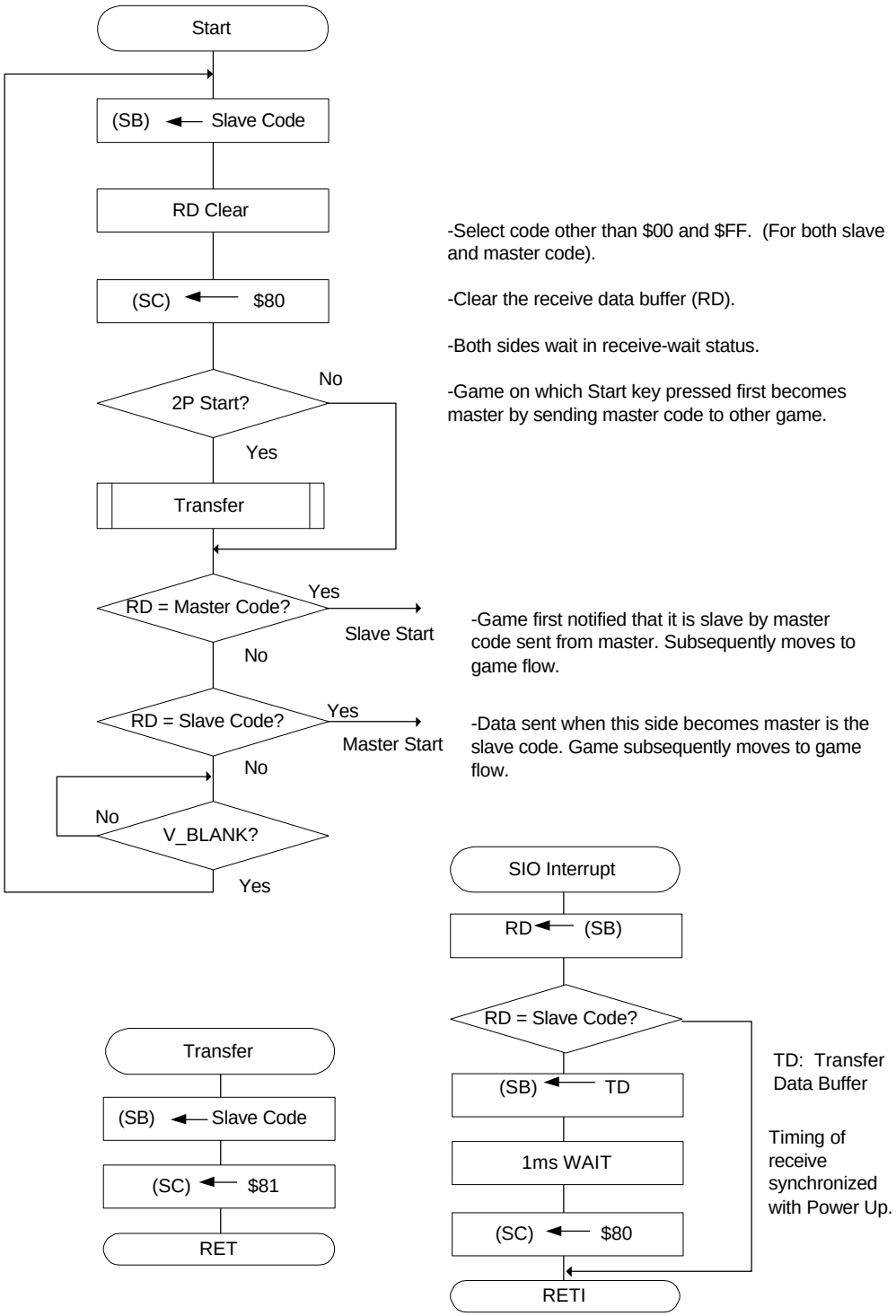


SIO Block Diagram

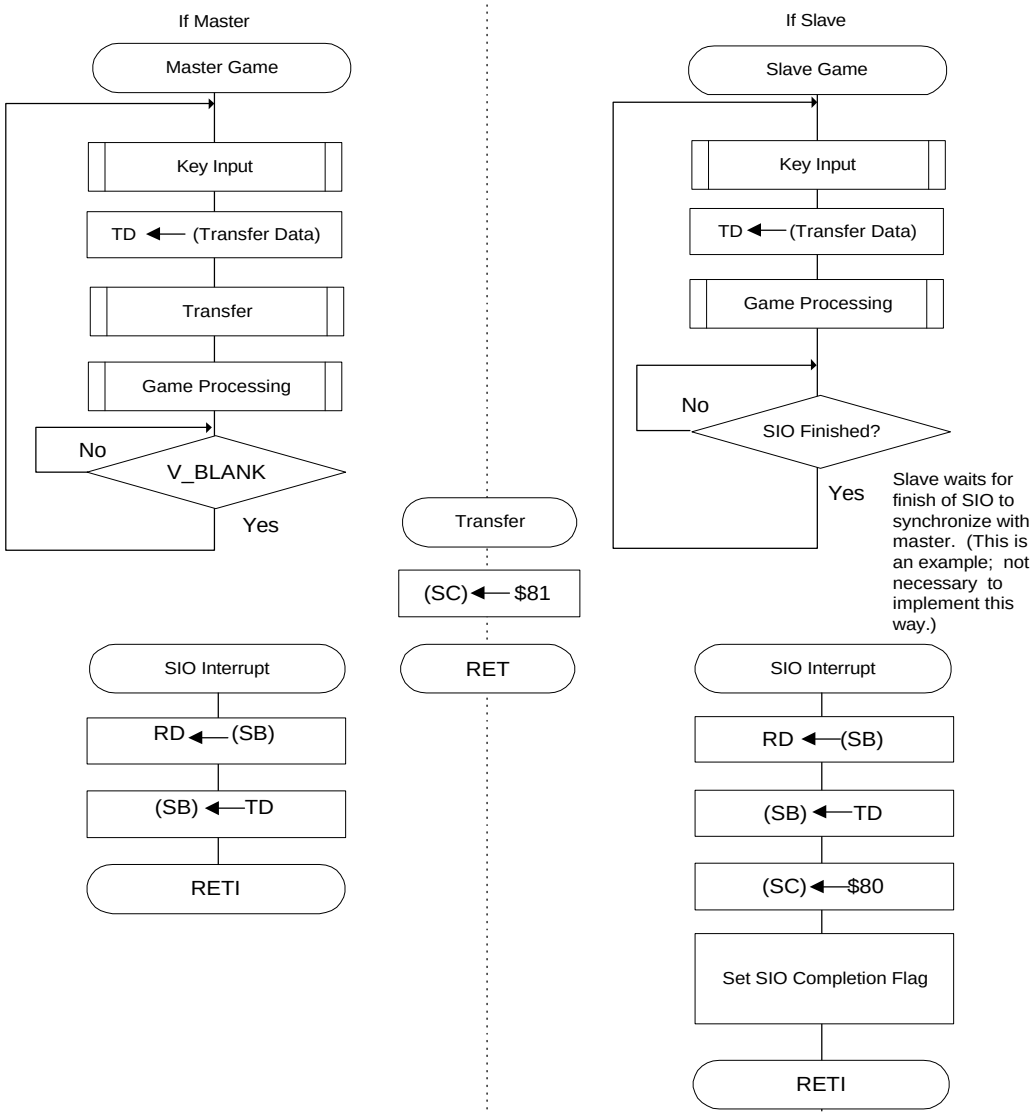


1.2.5.2 Serial Cable Communication: Reference flowchart

Flow until start of game



Flow after game start



Data subsequently sent by the master is placed in (SB) and then sent to the slave at the same time as the (SC) is set to $81. At exactly that same time, the master receives the slave data. An SIO interrupt is then set in the slave and, as the flowchart indicates, the slave sets the data to be sent to the master (current data).

Because the data sent from the slave are those loaded at the time of the previous interrupt, the data sent to the master are one step (one pass through the main program) behind the current slave data. Exactly the converse is true when this process is viewed from the perspective of the slave. An SIO interrupt is set in the master, and the master sets the data to be sent to the slave (current data). In this case, because the data sent from the master are those loaded at the time of the previous interrupt, the data sent to slave are one step (one pass through main program) behind the current master data. (*The data of the master and slave can be synchronized by setting the data for each back 1 pass.)

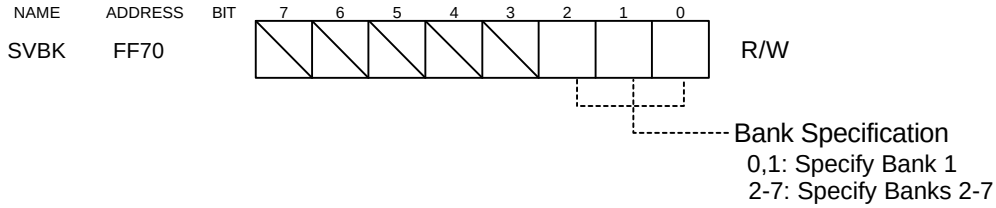
In the example, 1 byte is sent per frame. (This is not required.) If several bytes are sent continuously, a transmission interval longer than the processing time of other interrupts (e.g. V_BLANK) should be used (usually around 1 mS). The reason is that if an attempt is made to communicate with the slave during another interrupt, the slave cannot receive the data until after the interrupt is finished. If the next data is transmitted before the other interrupt is finished, the slave will be unable to receive the initial data of the transmission.

1.2.6 CPU Functions (CGB only)

This section describes CPU functions that can be used only with CGB. Functions that are identical in DMG and CGB are described in Section 1.2.4, *CPU Functions (Common to DMG/CGB①)*. For information on CPU functions enhanced in CGB, see Section 1.2.5, *CPU Functions (Common to DMG/CGB②)*.

1.2.6.1 Bank Register for Game Boy Working RAM

The 32 KB of Game Boy working RAM is divided into 8 banks of 4 KB each. The CPU memory space C000h-CFFFh is set to Bank 0, and the space D000h-DFFFh is switched between banks 1-7. Switching is performed using the lowest 3 bits of the bank register, SVBK. (If 0 is specified, Bank 1 is selected.)



Note: This register cannot be written to in DMG mode.

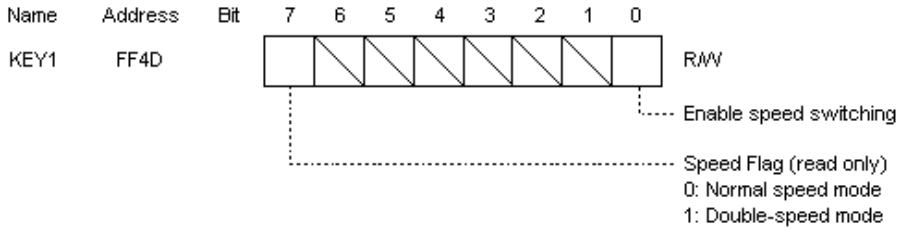
1.2.6.2 CPU Operating Speed

The speed of the CGB CPU can be changed to suit different purposes. In normal mode, each block operates at the same speed as with the DMG CPU. In double-speed mode, all blocks except the liquid crystal control circuit and the sound circuit operate at twice normal speed.

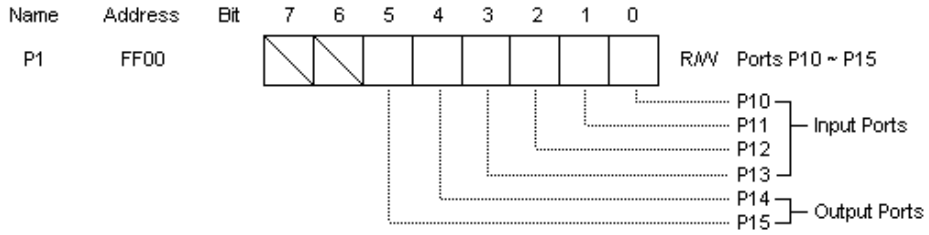
Normal mode: 1.05 MHz (CPU system clock)
Double-speed mode: 2.10 MHz (CPU system clock)

Switching the CPU Operating Speed

Immediately after the CGB CPU is reset (immediately after reset cancellation), it operates in normal mode. The CPU mode is switched by executing a STOP instruction with bit 0 of register Key 1 set to a value of 1. If this is done in normal mode, the CPU is switched to double-speed mode; otherwise it is switched to normal mode. Bit 0 of register Key 1 is automatically reset after the operating speed is switched. In addition, bit 7 of register Key 1 serves as the CPU speed flag, indicating the current CPU speed.



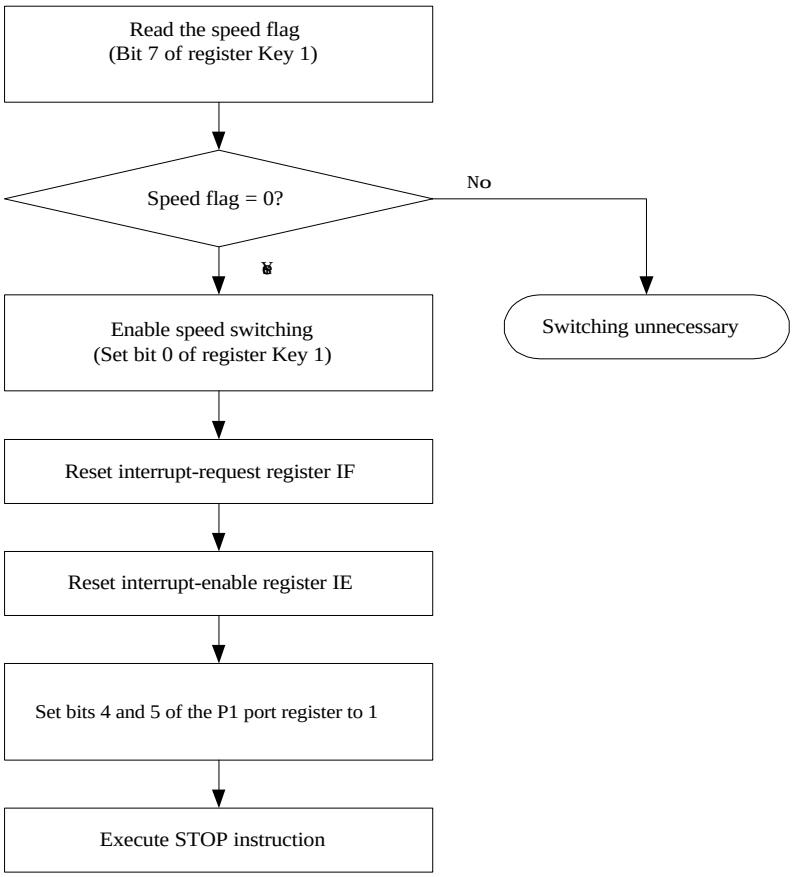
Note:	***When bit 0 of register Key 1 is set to 1, the standby function cannot be used. When using the standby function, always confirm that bit 0 of register Key 1 is set to 0. When switching the CPU speed, all interrupt-enable flags should be reset and a STOP instruction executed with bits 4 and 5 of the P1 port register set to 1, as with the standby function (STOP mode). When the CPU speed is switched, a return from STOP mode is automatic, so it is not necessary to generate a STOP mode cancellation. However, until the CPU speed has been changed and the system clock returns, bits 4 and 5 of the P1 port register should be made to hold the value 1.***



Approximately 16 ms is required to switch from normal to double-speed mode, and approximately 32 ms is needed to switch from double-speed to normal mode. In double-speed mode, the DIV register (FF04h) and the TIMA register (FF05h) both operate at double speed. Battery life is shorter in double-speed mode than in normal mode. The use of double-speed mode requires the corresponding mask ROM and MBC.

Flow of Switching (when switching to double-speed mode)

In case the CPU operating speed needed to be switched, the current speed should always be checked first using the speed flag (bit 7 of the KEY 1 register). This ensures that the speed will be switched to the intended speed.


Read the speed flag
(Bit 7 of register Key 1)
Speed flag = 0?
No
Switching unnecessary
Enable speed switching
(Set bit 0 of register Key 1)
Reset interrupt-request register IF
Reset interrupt-enable register IE
Set bits 4 and 5 of the P1 port register to 1
Execute STOP instruction

Switching Routine (example)

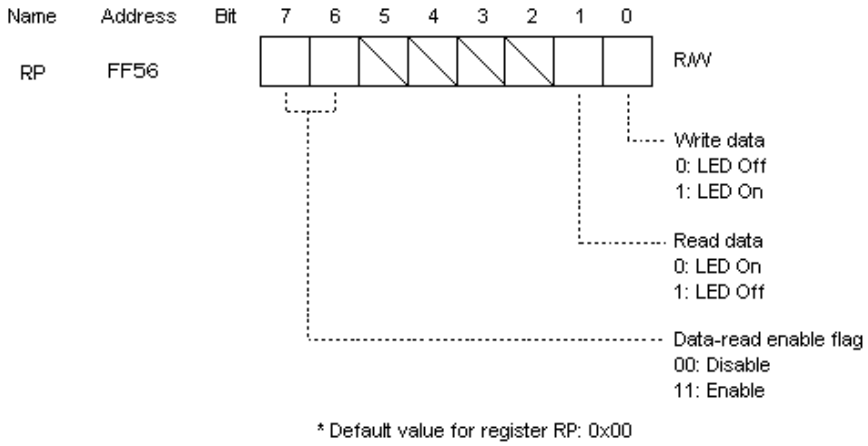
```
        LD      HL, KEY1
        BIT     7, (HL)
        JR      NZ, _NEXT
        SET     0, (HL)
        XOR     A
        LD      (IF), A
        LD      (IE), A
        LD      A, $30
        LD      (P1), A
        STOP
_NEXT
```

1.2.6.3 Infrared Communication

1.2.6.3.1 Port Register

The CGB system is equipped with an infrared communication function. An infrared signal can be output by writing data to bit 0 of RP register. A received infrared signal is latched internally in the CPU by positive edge of the system clock. (System clock goes to HIGH from LOW.) The latched data can be read beginning from bit 1 of RP register by setting bits 6 and 7 to 1.

Note: When data is not sent or received, always set the values of RP register to 0x00. This register cannot be written to in DMG mode.



1.2.6.3.2 Controlling Infrared Communication

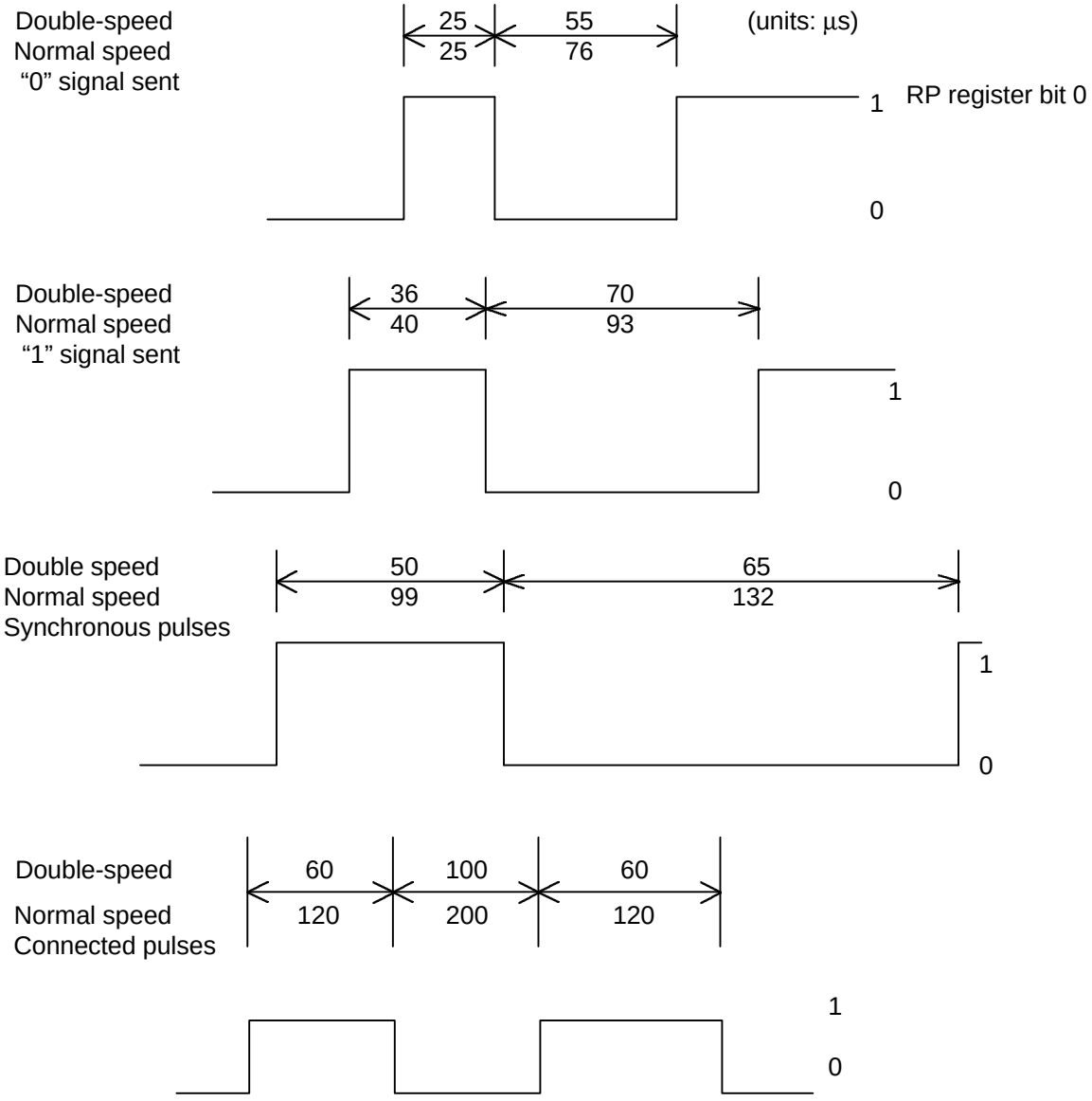
Sender:
Setting bit 0 of the RP register to 1 causes the LED to emit light; setting it to 0 turns off the LED.

Receiver:
If the photo transistor detects infrared light, bit 1 of the RP register is set to 0; if no infrared light is detected, this bit is set to 1.

1.2.6.3.3 Basic Format

When the receiver recognizes the unmodified signal from the sender as a logical value of 1 or 0, the receiver actually cannot distinguish between the continuous transmission of 1s and the absence of received infrared light. The status of the receiver is identical under these conditions. Consequently, to ensure proper data transmission from sender to receiver in Game Boy Color infrared communication, signals are distinguished by the size of the interval between the rising edge of the pulse of one received signal to the rising edge of the subsequent received signal.

The following illustrates signals from a sender.



Scatter in the source oscillation of Game Boy Color produces slight individual differences.

1.2.6.3.4 Preparing for Data Transmission and Reception

To use infrared communication, data reception must be enabled by setting bits 6 and 7 of Game Boy Color RP register to 1. However, even with both of these bits set to 1, data cannot immediately be received. After setting bits 6 and 7 to 1, at least 50 ms should be allowed to pass before using the infrared port.

1.2.6.3.5 Transmitted Data

When data is transmitted and received, it is transmitted in packets. Each packet comprises the 4 parts shown below, and each part is sandwiched between synchronous pulses. For more information, see Section 1.2.6.3.7, Details of Data Transmission and Reception.

The data that comprises a packet is transmitted 1 bit at a time beginning from the MSB.

Transmission Packet

Connector	Header	Data	Checksum

Connector:
Signal that implements an infrared communication connection between two Game Boy Color machines. This is always required in the initial packet. When the receiver receives the connector and recognizes it as a connecting pulse, the receiver returns the same pulse to the sender. The sender then determines whether this signal is a normal connecting pulse. If it is not recognized as a normal pulse, transmission is interrupted at this stage.

Header:
Data indicating the type of data being sent and the total number of bytes.

Byte 1: Communication command
5Ah: transmission of raw data
At present, any value other than 5Ah causes an error.
(To be used for by other devices in future)

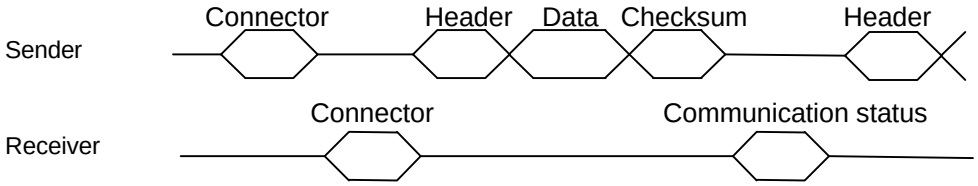
Byte 2: Total number of data in data portion of the packet
01h-FFh: Number of data
00h: Indicates completion of communication to receiver.

Data:
The transmitted data itself. Maximum of 255 bytes.
There are no data if completion of communication is indicated to the receiver.
(The data portion of the packet consists only of a synchronous pulse.)

Checksum:
2 bytes of data consisting of the sum of the header and all data in the data portion of the packet. Following this, the communication status is returned from receiver to sender.

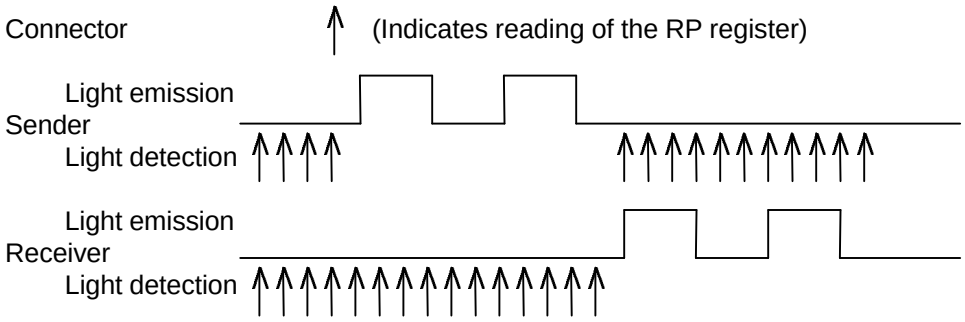
1.2.6.3.6 Flow of Data Transmission and Reception

When data is transmitted and received, both Game Boy Color units are first placed in receive status. The one with the send indicator is then designated as the sender, and the other one is designated as the receiver. The flow of data transmission is shown below.



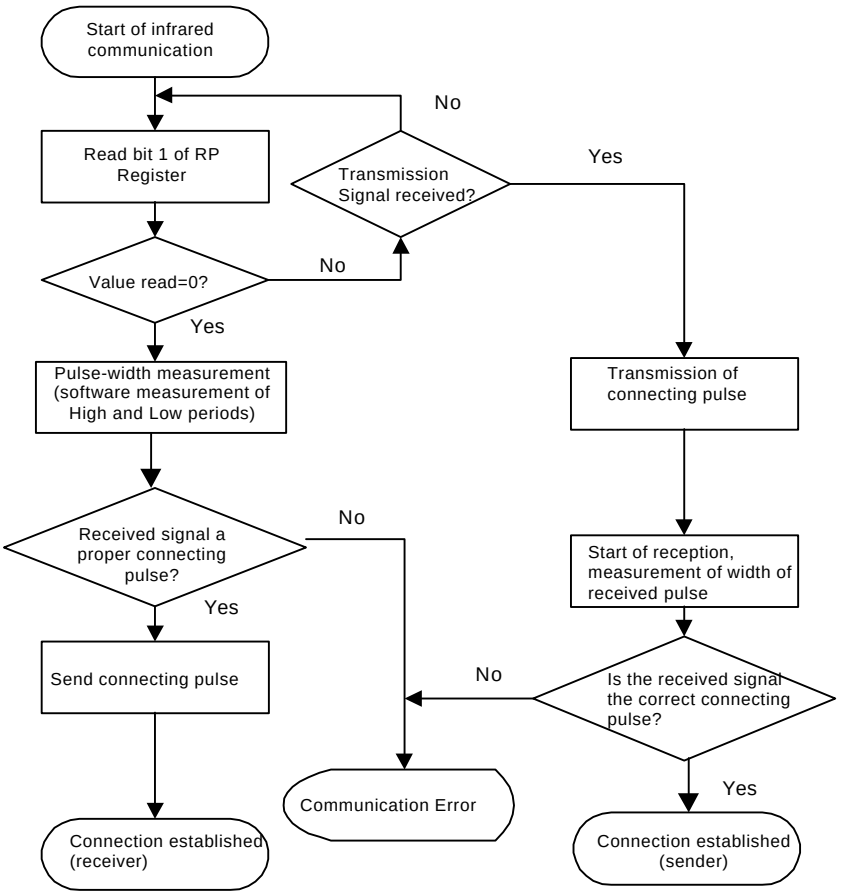
1 Sender transmits connecting pulse.
2 The receiver calculates the width of the received connecting pulse. If the value is correct, the receiver returns the same connecting pulse to the sender.
3 The sender calculates the width of the connecting pulse returned by the receiver. If the value is correct, the sender determines that a connection has been properly established.
4 The header is transmitted.
5 The data is transmitted.
6 The checksum is transmitted.
7 The receiver returns the communication status to the sender.
8 When communication is complete, the header of the subsequently transmitted packet is set to 00h + 00h.

1.2.6.3.7 Details of Data Transmission and Reception

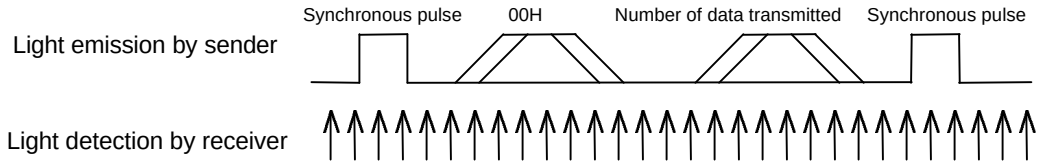


The two Game Boy Color machines perform initial data reception, then the one designated as the sender (e.g., by operations such as pressing button A) begins transmission.

The following illustrates the flow for implementing a connection.

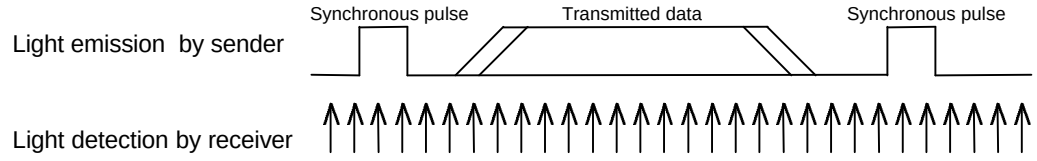


Header



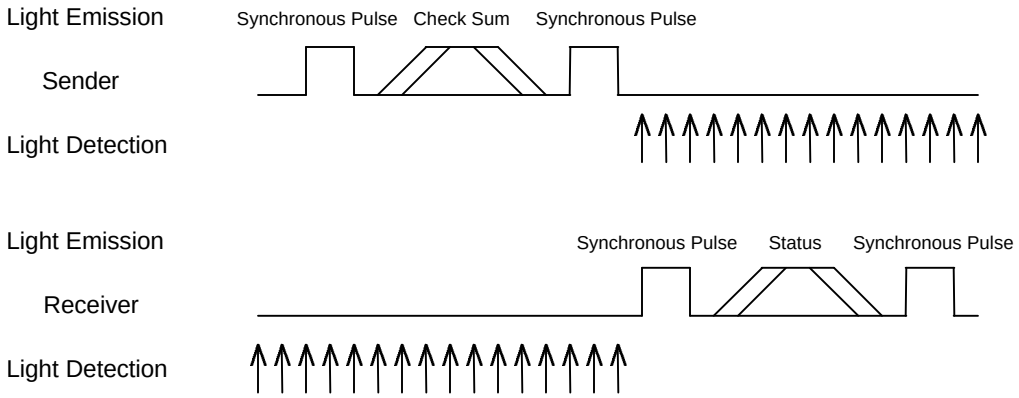
One byte indicating the data type and 1 byte indicating the number of transmitted data are sandwiched between synchronous pulses.

Data



Between 1 and 255 bytes of transmitted data are sandwiched between synchronous pulses.

Checksum



A 2-byte checksum consisting of the sum of the header and transmitted data is sandwiched between synchronous pulses. The receiver uses the checksum to determine whether the transmission was performed properly and notifies the sender of the results of communication status.

The following section describes the details of communication status determination.

1.2.6.3.8 Communication Status

8Bh : Communication OK
04h : Checksum error

The results of the checksum calculated by the receiver do not agree with the checksum sent by the sender.

In the following cases, the communication status cannot be returned to the sender even if an error is generated during communication (no response from receiver).

- The wrong communication protocol is used.
- Data is transmitted using the wrong pulse width.
- One of the Game Boy Color units is operating in double-speed mode and the other is operating in normal mode.
- Communication is affected by sunlight or obstruction of the signal light.

1.2.6.3.9 Communication Error Processing

If an error described above in Communication Status is generated, the following error codes are returned by subroutine.

Error Code	Error Description
04h	Checksum error (same for sender and receiver): The results of the checksum calculated by the receiver and the checksum sent by the sender do not agree.
10h	Pulse width error: Generated by the receiver when the width of the pulse of the signal sent by the sender is too wide or narrow. Generated by the sender when the width of the pulse of the signal sent by the receiver is too wide or narrow.
20h	Communication error: Communication prevented by other causes. The subroutine provided by Nintendo treats as an error the case when the data value of the second byte of the received header exceeds the number of data items to be received, as determined beforehand by the receiver. The routine also generates an error if the communication command value of byte 1 of the header is not 5A.

1.2.6.3.10 Usage Notes

When programming use of the infrared port, please note the following.

- When transmitting more than 256 bytes of data, ensure that the receiver keeps track of which packet number is being received. When a communication error (status not returned even though data was received) is generated, the sender will re-send the data, and the receiver may lose track of the packet number (see code 04h in previous section).
- The sender is prone to entering an endless loop when the packet signifying transmission completion is received. Therefore, the receiver should remain in receive status for approximately 300 μs after returning the status (see code 10h in previous section).
- Depending on the power reserve of the battery, infrared communication may cause a sudden drop in battery voltage and a complete loss of power.
- Ensure that the speed of the two communicating Game Boy Color machines is the same (both double-speed or both normal speed during communication).
- Noise can be heard from the speaker and headphones during communication, but this does not indicate a problem with the hardware.
- Be careful that malfunctions/lock-ups do not occur when infrared communication signals are input from other game software and devices. Use particular care when using the same subroutine to communicate between various types of games because malfunctions/lock-ups are especially likely to occur in such cases. (Before performing data communication, confirm that the other hardware involved in the transmission is using the same game. This can be accomplished by means such as exchanging a unique key code.)
- Though very rare, it is possible that at the final communication stage, one Game Boy will terminate normally and the other abnormally due to an unexpected external disturbance.

The following are items to note when using an infrared communication subroutine other than that provided by Nintendo.

- Ensure that error-handling is implemented to prevent the program from entering an endless loop when communication is interrupted by sunlight or obstruction of the signal light.
- To reduce power consumption, use a maximum infrared LED emission pulse duration of 150 μs and a duty ratio of approximately 1/2.
- Do not leave the infrared LED or photo transistor(Amplifier and Read Enable) ON when not using infrared communication.

1.2.6.3.11 Specifications

1) Communication Speed
   Normal-speed mode: approximately 7.5 Kbps
   Double-speed mode: approximately 9.5 Kbps
2) Communication distances: Minimum, 10 cm, Typical, 15 cm
3) Recommended directional angle: approximately ± 15º

CHAPTER 2 DISPLAY FUNCTIONS.....49

2.1 General Display Functions.....49

2.1.1 Character Composition.....49
2.1.2 LCD Display RAM.....50
2.1.3 Character RAM.....51
2.1.4 BG Display.....55
2.1.5 LCD Screen.....57
2.1.6 LCD Display Registers.....58
2.1.7 OAM Registers.....63
2.1.8 DMA Registers.....65
2.1.9 OBJ Display Priority.....71

2.2 LCD COLOR DISPLAY (CGB ONLY).....73

2.2.1 Color Palettes.....73
2.2.2 Color Palette Composition.....74
2.2.3 Writing Data to a Color Palette.....74
2.2.4 Overlapping OBJ and BG.....76
2.2.5 Display Using Earlier DMG Software (DMG mode).....77

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 2 DISPLAY FUNCTIONS

2.1 General Display Functions

2.1.1 Character Composition

- The basic character size is an 8 x 8-dot composition.
- With characters of the basic size:
  - 128 OBJ-only characters are available (256 with CGB)
  - 128 BG-only characters are available (256 with CGB)
  - 128 characters can be registered both as OBJ and BG characters (256 with CGB)
- On DMG, characters can be represented using 4 shades of gray (including transparent). On CGB, characters can be represented using 32 shades for each color of RGB.
- The basic character size can be switched to an 8 x 16-dot composition for OBJ characters only. In this case, however, only even-numbered character codes can be specified. Even if an odd-numbered character code is specified, the display will be the same as that seen with an even-numbered code.
- Up to 40 OBJ characters can be displayed in a single screen, and up to 10 characters can be displayed on each horizontal line.
- The display data for OBJ characters are as follows:
  - y-axis coordinate
  - x-axis coordinate
  - Character code
  - Attribute data
- Data are written to OAM from working RAM by DMA transfer.
- OBJ characters are automatically displayed to the screen using the data written to OAM.
- Data specification ranges for OBJ characters:
  - 00 ≤ character code ≤ FFh
  - 00 ≤ X ≤ FFh
  - 00 ≤ Y ≤ FFh

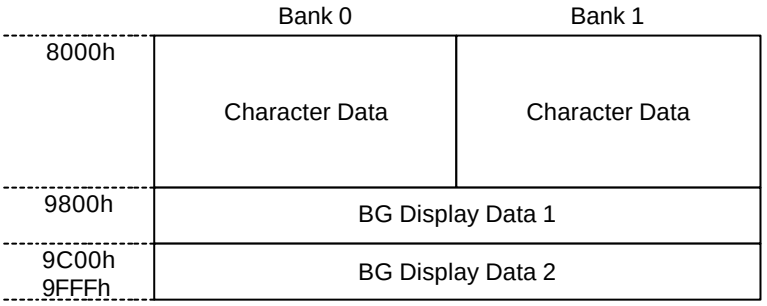
2.1.2 LCD Display RAM

The DMG CPU has 8 KB (64 Kbits) of built-in LCD display RAM.

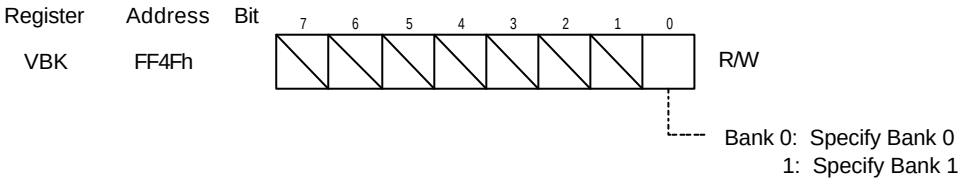
In CGB, 16 KB of memory can be joined in the 8 MB (64 Kbit) memory area (8000h-9FFFh) by bank switching using the register VBK (FF4Fh). Bank switching is used exclusively in CGB and cannot be used in DMG mode.

- Mapping of LCD Display RAM

The 16 MB of memory in CGB is partitioned into 2 x 8 KB by register VBK.



- Bank Register (CGB) for LCD Display RAM



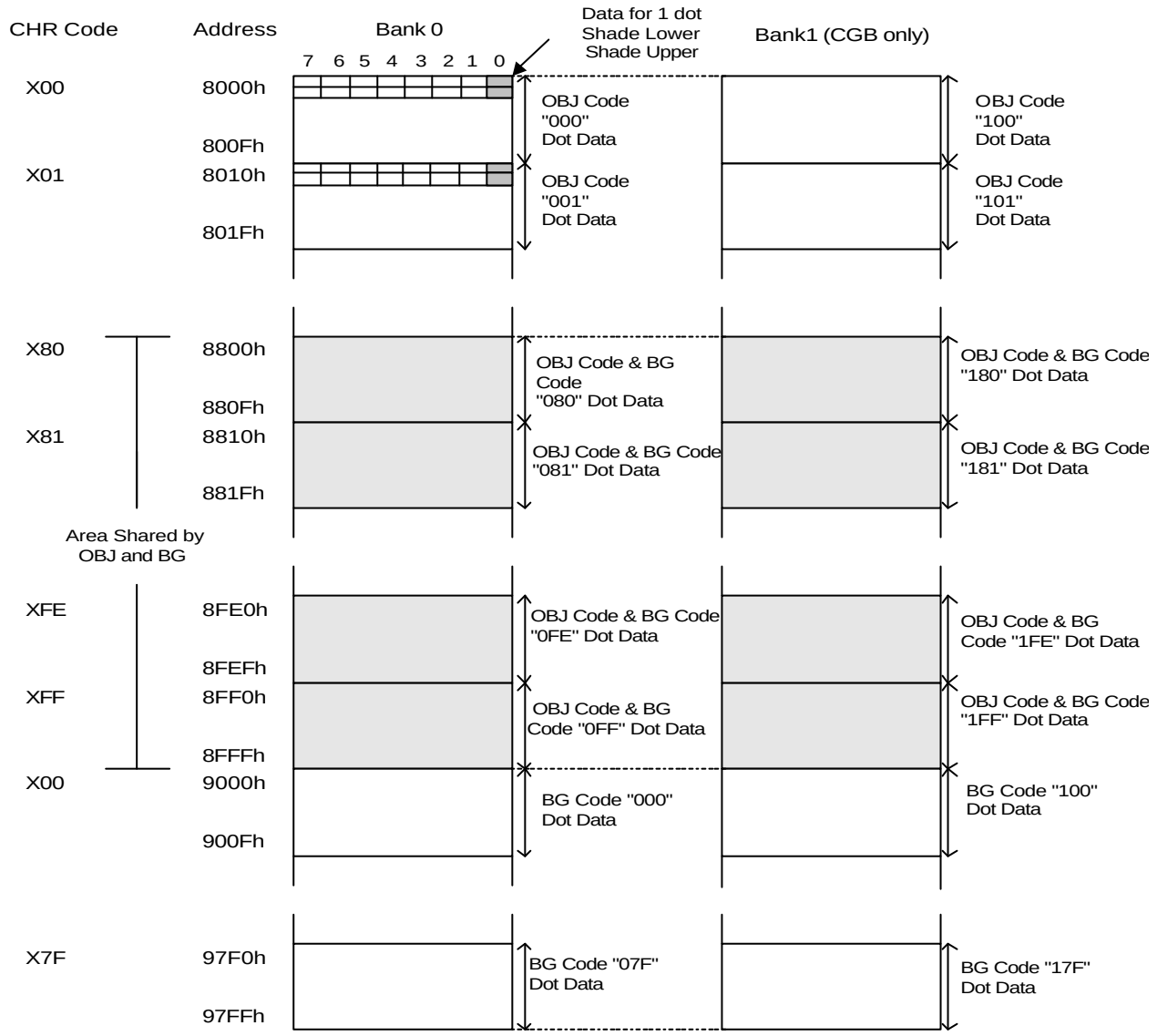
Bank 0 is selected immediately after cancellation of a reset signal.

This function is available only in CGB. In DMG mode, bit 0 is forcibly set 0, and its value cannot be changed to 1.

2.1.3 Character RAM

- Character data can be written to the 6144 bytes from 8000h to 97FFh.
- The area from 8000h to 8FFFh is allocated for OBJ character data storage.
- The register LCDC can be used to select either 8000h-8FFFh or 8800h-97FFh as the area for storing BG and window character data.
- If the BG character data are allocated to 8000h-8FFFh, these data share an area with OBJ data, and the character dot data that correspond to the CHR codes also are the same.
- By means of bank switching, CGB can store twice the amount of character data in LCD display RAM that DMG can store. In this case, both Bank 1 and Bank 0 have the same mapping as the area in DMG.

Character Code Mapping



With BG character data allocated to 8800h-97FFh:

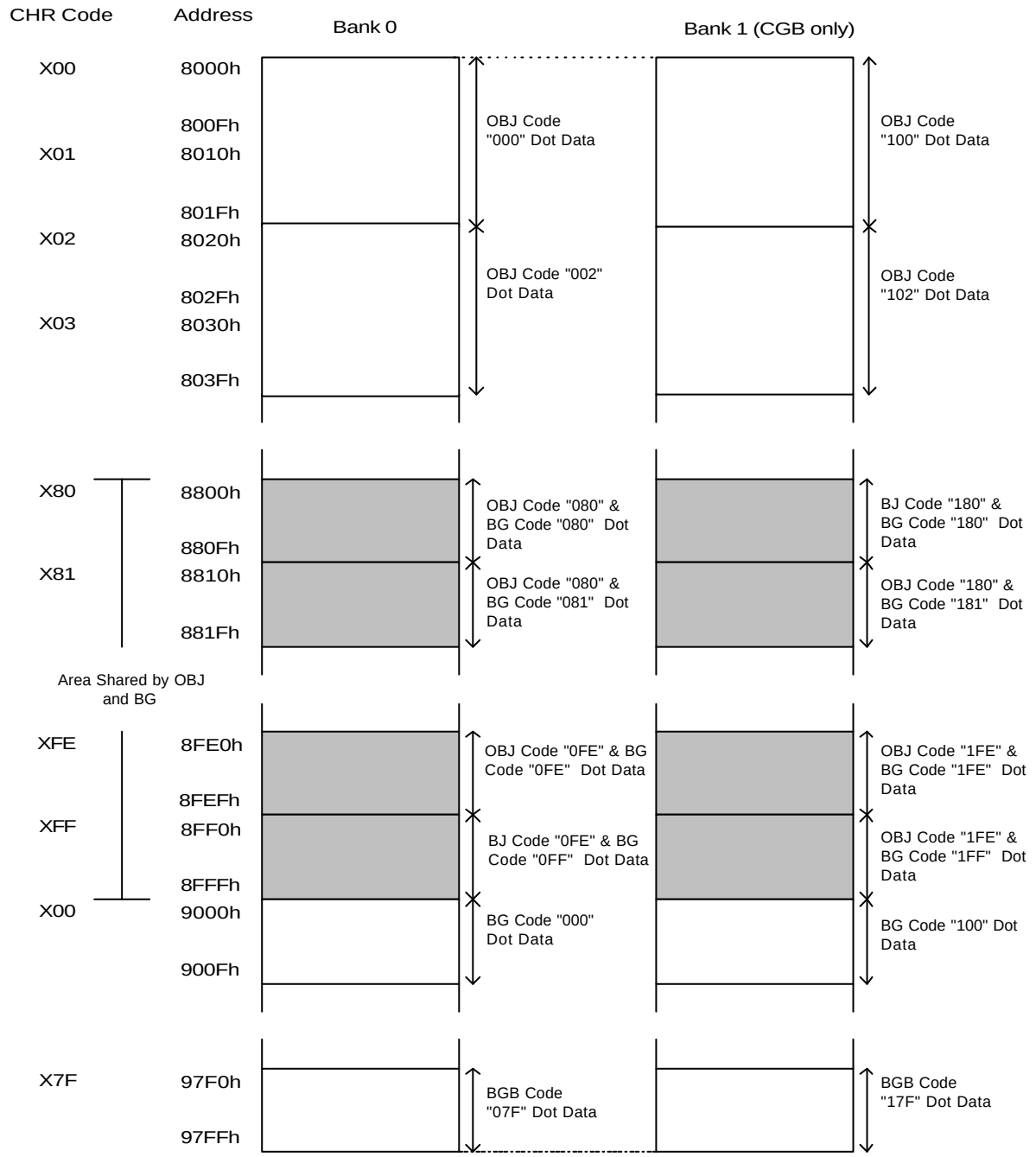
- The case of 8 x 8 dots/block for both BG and OBJ:

  CHR Codes:

<DMG>	<CGB>
OBJ: 256 x 1	OBJ: 256 x 2
BG: 256 x 1	BG: 256 x 2

Note: ***Because bank switching is not available in DMG mode, Bank 1 on the right side of the figure is not available in this mode.***

8 x 16 dots/block (OBJ) and 8 x 8 dots/block (BG):



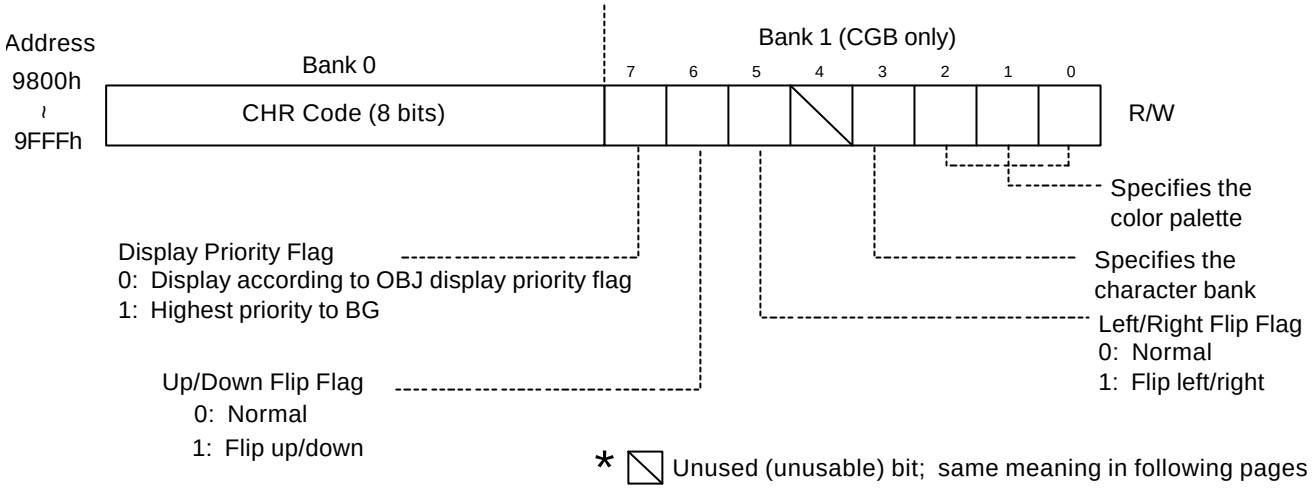
CHR Codes:

<DMG>		<CGB>	
OBJ:	128 x 1	OBJ:	128 x 2
BG:	256 x 1	BG:	256 x 2

If BG character data are allocated to 8000h-8FFFh, these data share an area with OBJ data, and the dot data that correspond to the CHR codes also are the same.

Note: ***Because bank switching is not available in DMG mode, Bank 1 on the right side of the figure is not present in this mode.***

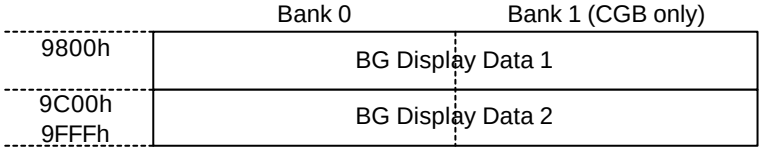
2.1.4 BG Display



Two screens of BG display can be held, Data 1 or Data 2.

Whether the BG display data are allocated to 9800h-9BFFh or to 9C00h-9FFFh is determined by bit 3 of the LCDC register (FF40h).

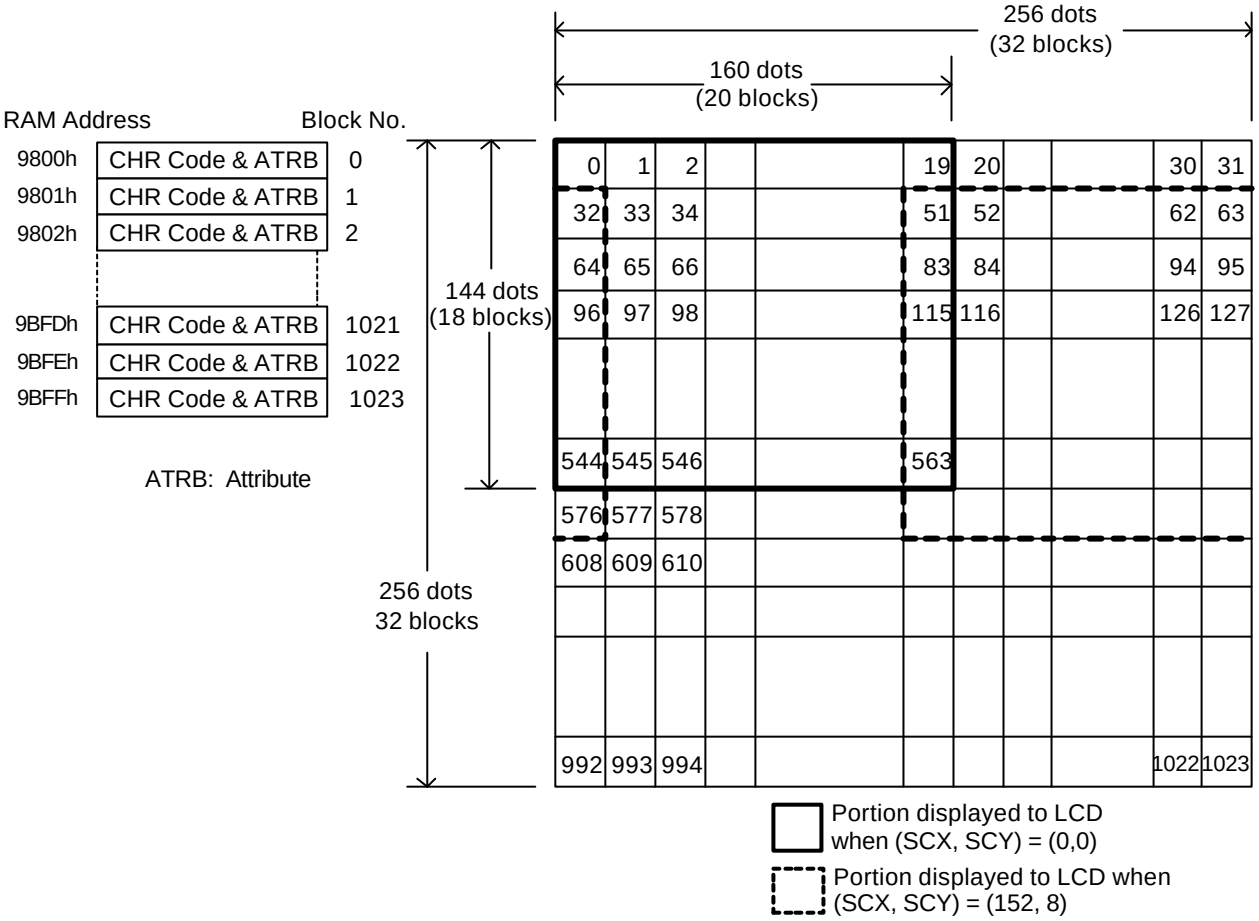
Because bank switching is not available in DMG mode, Bank 1 on the right side of the figure is not present in this mode.



Data for 32 x 32 character codes (256 x 256 dots) can be specified from 9800h or 9C00h as BG display data. Of these, data for 20 x 18 character codes (160 x 144 dots) are displayed to the LCD screen.

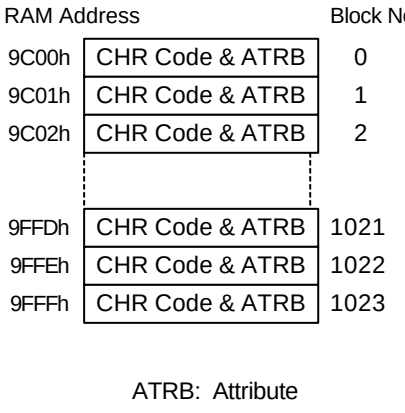
The screen can be scrolled vertically or horizontally one dot at a time by changing the values of scroll registers SCX and SCY.

1) With BG display data allocated to 9800h-9BFFh:


256 dots
(32 blocks)
160 dots
(20 blocks)
RAM Address
Block No.
9800h CHR Code & ATRB 0
9801h CHR Code & ATRB 1
9802h CHR Code & ATRB 2
9BFDh CHR Code & ATRB 1021
9BFEh CHR Code & ATRB 1022
9BFFh CHR Code & ATRB 1023
ATRB: Attribute
144 dots
(18 blocks)
256 dots
32 blocks
0 1 2 19 20 30 31
32 33 34 51 52 62 63
64 65 66 83 84 94 95
96 97 98 115 116 126 127
544 545 546 563
576 577 578
608 609 610
992 993 994 1022 1023
Portion displayed to LCD
when (SCX, SCY) = (0,0)
Portion displayed to LCD when
(SCX, SCY) = (152, 8)

Note: Attributes specified only with CGB

2) With BG display data allocated to 9C00h-9FFFh:


RAM Address
Block No.
9C00h CHR Code & ATRB 0
9C01h CHR Code & ATRB 1
9C02h CHR Code & ATRB 2
9FFDh CHR Code & ATRB 1021
9FFEh CHR Code & ATRB 1022
9FFFh CHR Code & ATRB 1023
ATRB: Attribute

Correspondence between LCD screen and block numbers as shown in preceding figure.

Note: Attributes specified only with CGB.

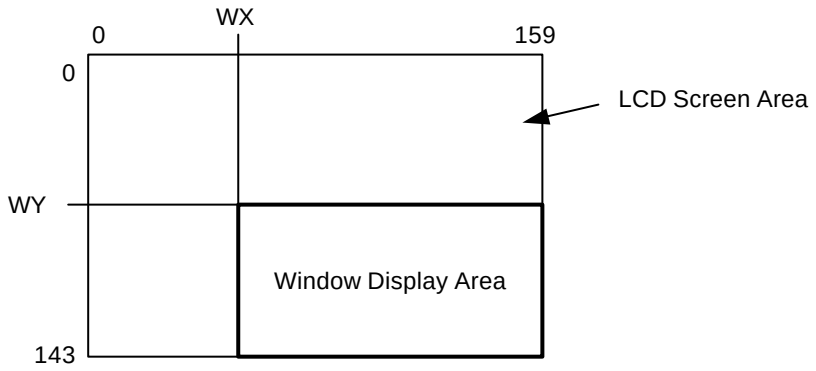
2.1.5 LCD Screen

- Window Display

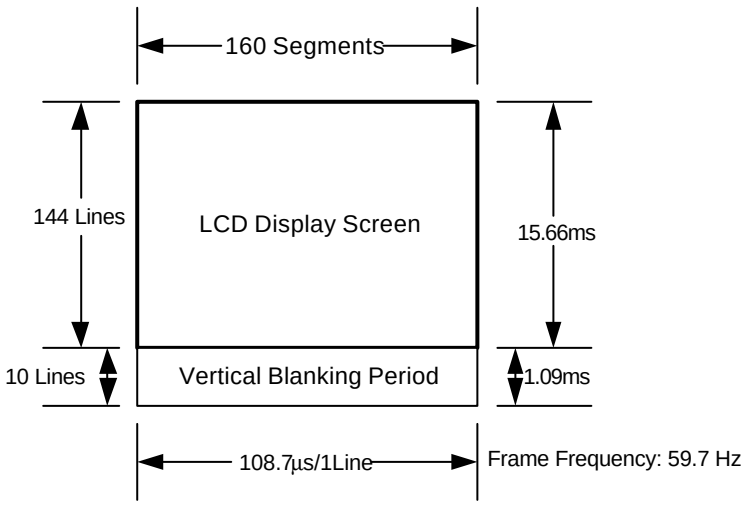
Specifying a position on the LCD screen using registers WX and WY causes the window to open downward and rightward beginning from that position.

Window display data also can be specified as character codes, beginning from 9800h or 9C00h in external SRAM.

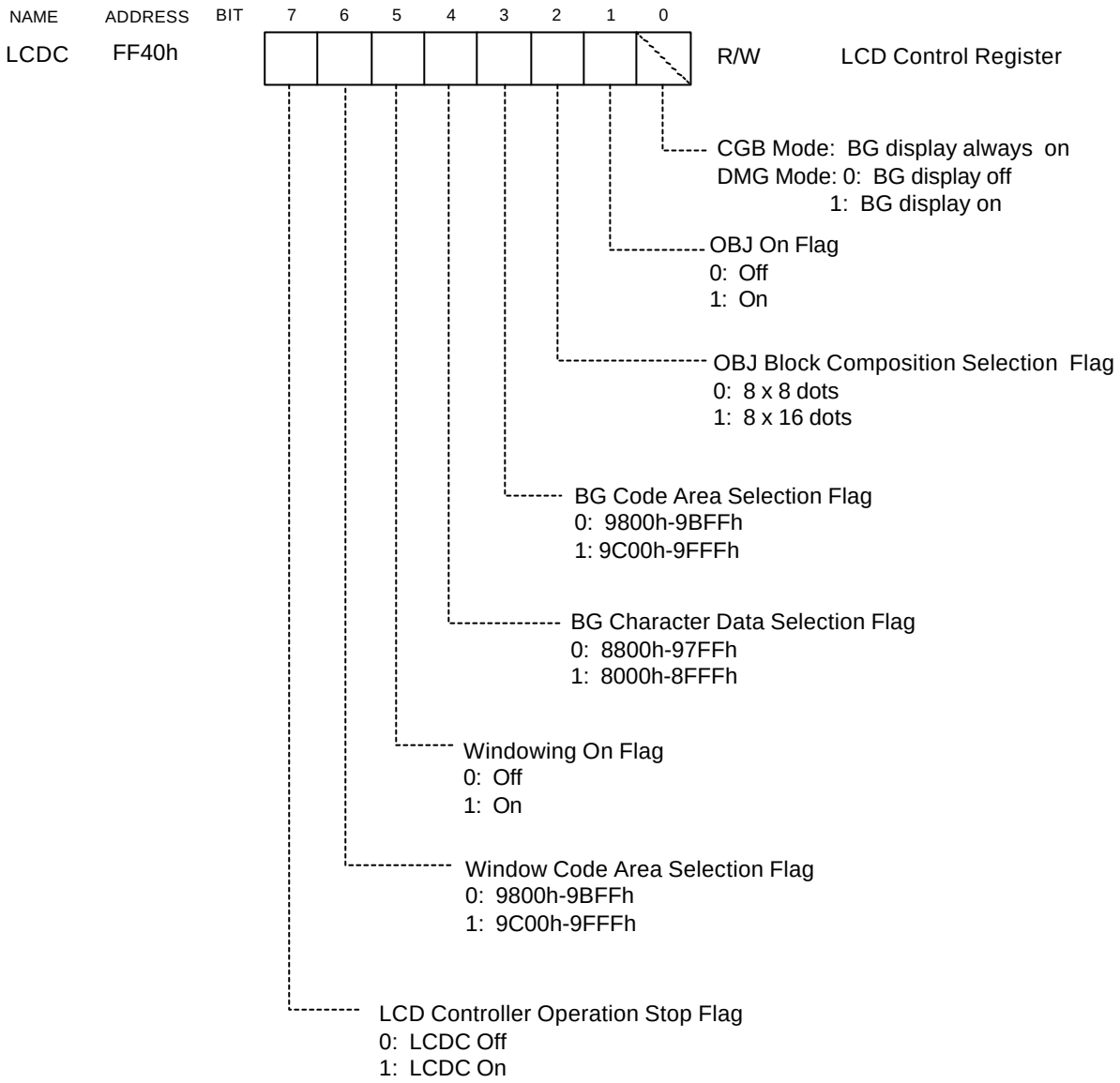
OBJ character data are displayed in the window in the same way as the BG screen.



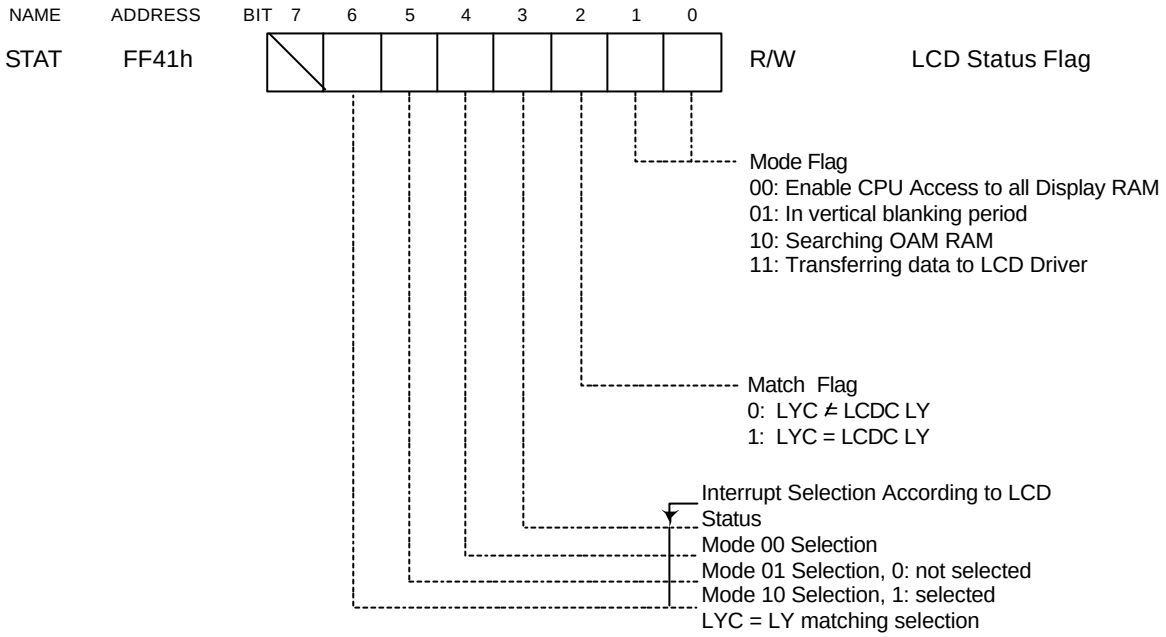
- Screen Timing



2.1.6 LCD Display Registers



* In CGB, the liquid crystal protection circuit functions when the LCDC is turned on. Consequently, a white screen is displayed for up to 2 frames. In DMG, the LCDC should be off during vertical blanking periods.


NAME
ADDRESS
BIT 7 6 5 4 3 2 1 0
STAT
FF41h
R/W
LCD Status Flag
Mode Flag
00: Enable CPU Access to all Display RAM
01: In vertical blanking period
10: Searching OAM RAM
11: Transferring data to LCD Driver
Match Flag
0: LYC ≠ LCDC LY
1: LYC = LCDC LY
Interrupt Selection According to LCD Status
Mode 00 Selection
Mode 01 Selection, 0: not selected
Mode 10 Selection, 1: selected
LYC = LY matching selection

STAT indicates the current status of the LCD controller.

Mode 00: A flag value of 1 represents a horizontal blanking period and means that the CPU has access to display RAM (8000h-9FFFh).

When the value of the flag is 0, display RAM is in use by the LCD controller.

Mode 01: A flag value of 1 indicates a vertical blanking period and means that the CPU has access (approximately 1 ms) to display RAM (8000h-9FFFh).

Mode 10: A flag value of 1 means that OAM (FE00h-FE90h) is being used by the LCD controller and is inaccessible by the CPU.

Mode 11: A flag value of 1 means that the LCD controller is using 0AM (FE00h-FE90h) and display RAM (8000h-9FFFh). The CPU cannot access either of these areas.

In addition, the register allows selection of 1 of the 4 types of interrupts from the LCD controller. Executing a write instruction for the match flag resets that flag but does not change the mode flag.

NAME	ADDRESS	BIT 7	6	5	4	3	2	1	0		
SCY	FF42h									R/W	Scroll Y 00~FF
NAME	ADDRESS	BIT 7	6	5	4	3	2	1	0		
SCX	FF43h									R/W	Scroll X 00~FF

Changing the values of SCY and SCX scrolls the BG screen vertically and horizontally one bit at a time.

NAME	ADDRESS	BIT 7	6	5	4	3	2	1	0		
LY	FF44h									R	LCDC y-coordinate

LY indicates which line of data is currently being transferred to the LCD driver. LY takes a value of 0-153, with 144-153 representing the vertical blanking period.

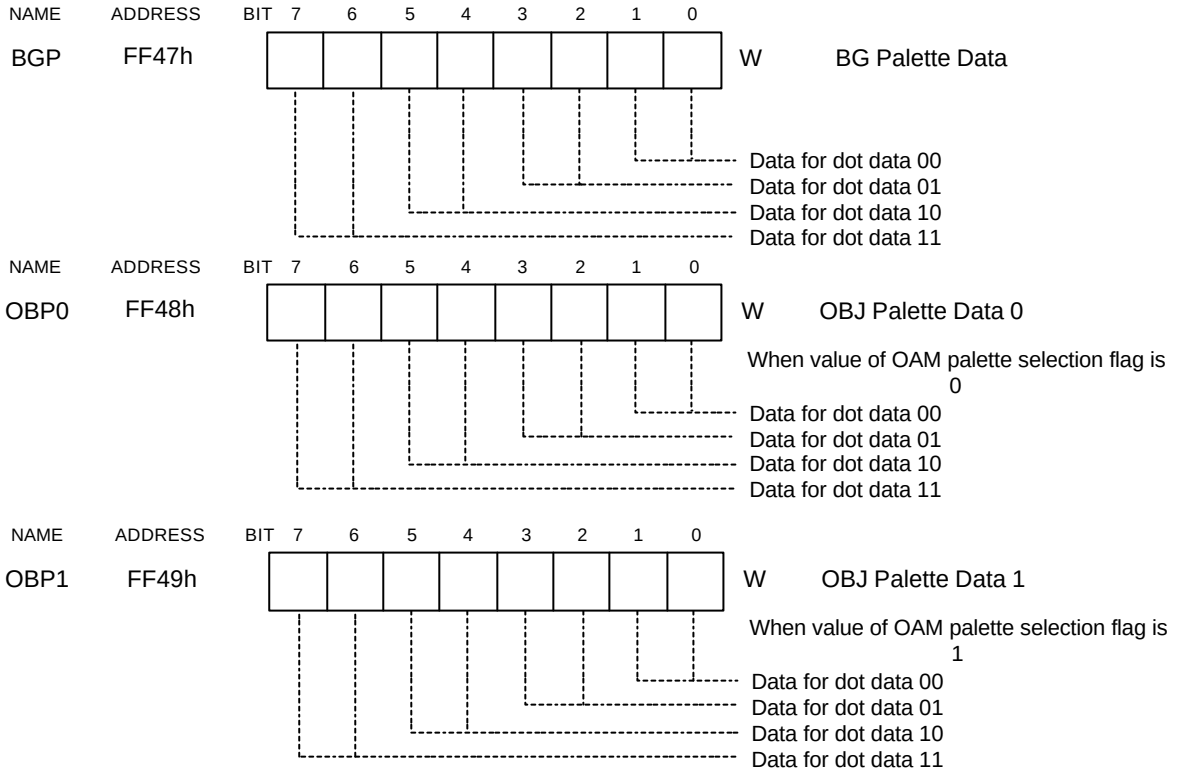
When the value of bit 7 of the LCDC register is 1, writing 1 to this again does not change the value of register LY.

Writing a value of 0 to bit 7 of the LCDC register when its value is 1 stops the LCD controller, and the value of register LY immediately becomes 0. (Note: Values should not be written to the register during screen display.)

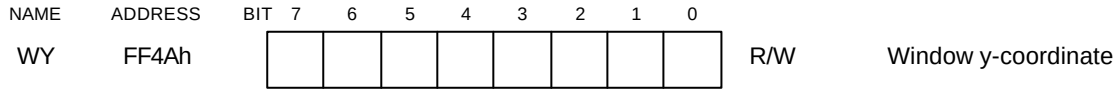
NAME	ADDRESS	BIT 7	6	5	4	3	2	1	0		
LYC	FF45h									R/W	LY Compare

Register LYC is compared with register LY. If they match, the Matchflag of the STAT register is set.

Note: ***The following 3 registers (BGP, OBP0, and OBP1) are valid in DMG and CGB modes. For information of CGB color palette settings, see section 2.2, LCD Color Display.***

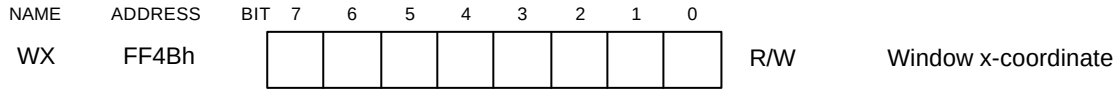

NAME ADDRESS BIT 7 6 5 4 3 2 1 0
BGP FF47h
W BG Palette Data
Data for dot data 00
Data for dot data 01
Data for dot data 10
Data for dot data 11
NAME ADDRESS BIT 7 6 5 4 3 2 1 0
OBP0 FF48h
W OBJ Palette Data 0
When value of OAM palette selection flag is 0
Data for dot data 00
Data for dot data 01
Data for dot data 10
Data for dot data 11
NAME ADDRESS BIT 7 6 5 4 3 2 1 0
OBP1 FF49h
W OBJ Palette Data 1
When value of OAM palette selection flag is 1
Data for dot data 00
Data for dot data 01
Data for dot data 10
Data for dot data 11

The grayscales (2 bit) for the character dot data are converted by the palette data (BG: register BGP; OBJ: OBP0 or OBP1) and output to the LCD driver as data representing 4 shades (including transparent).



$0 \leq WY \leq 143$

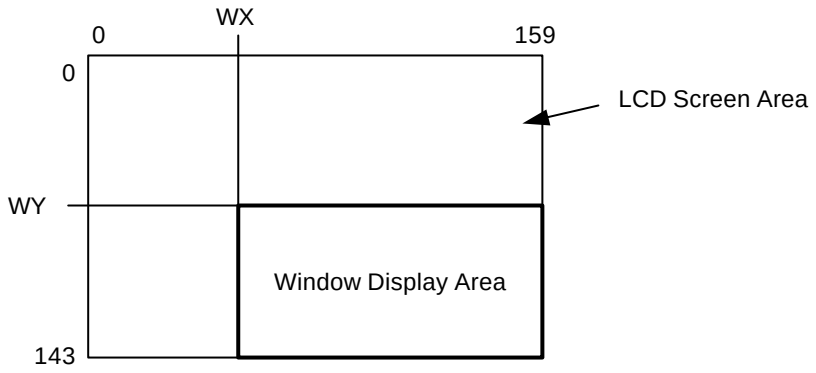
With WY = 0, the window is displayed from the top edge of the LCD screen.



$7 \leq WX \leq 166$

With WX = 7, the window is displayed from the left edge of the LCD screen.

Values of 0-6 should not be specified for WX.



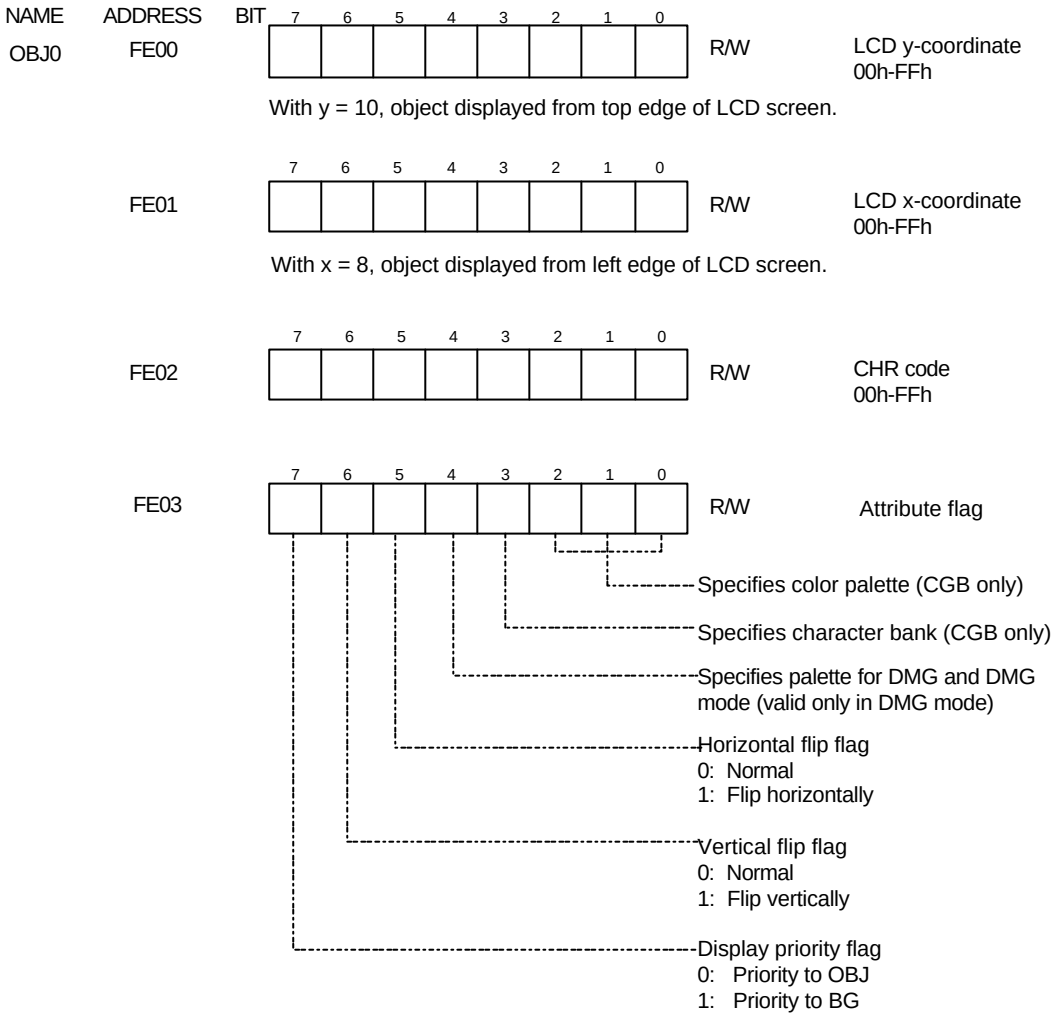
OBJ characters are displayed in the same manner in the window as on BG.

2.1.7 OAM Registers

OBJ (Object)

- Data for 40 objects (OBJ) can be loaded into internal OAM RAM in the CPU (FE00h-FE9Fh), and 40 objects can be displayed to the LCD. Up to 10 objects can be displayed on the same Y line.
- Each object consists of a y-coordinate (8 bits), x-coordinate (8 bits), and CHR code (8 bits) and specifications for BG and OBJ display priority (1 bit), vertical flip (1bit), horizontal flip (1 bit), DMG-mode palette, (1 bit), character bank (1bit), and color palette (3 bits), for a total of 32 bits.
- An 8 x 8- or 8 x 16-bit block composition can be specified for an OBJ using bit 2 of the LCDC register. With an 8 x 16-bit composition, the CHR code is specifed as an even number, as in DMG.

OAM Register



OBJ1-OBJ39 have the same composition as OBJ0.

Note: ***In DMG mode, the lower 4 bits of the attribute flag are invalid; only the flags in the upper 4 bits starting from the palette flag are valid.***

2.1.8 DMA Registers

2.1.8.1 DMA Transfers in DMG

DMA transfers of 40 x 32 bits of data can be performed from the RAM area (8000h-DFFFh) to OAM (FE00h-FE9Fh). The transfer time is 160 μs.

Note that in DMG, data cannot be transferred by DMA from RAM area 0000h-7FFFh.

The starting address of a DMA transfer can be specified as 8000h-DFFFh in increments of 100h.

Note that the method used for transfers from 8000h-9FFFh (display RAM) is different from that used for transfers from other addresses.

Example 1

The following example shows how to perform a DMA transfer of 40 x 32 bits from the expansion RAM area (C000h-C09Fh) to OAM (FE00h-FE9Fh).

During DMA, the CPU is run using the internal RAM area (FF80h-FFFEh) to prevent external bus conflicts.

1. The program writes the following instructions to internal RAM (FF80h-FFFEh):

Address	Machine Code	Label	Instruction		Comment
FF80	3E C0		LD	A, 0C0H	
	E0 46		LD	(DMA), A	;C000-C09F→OAM
	3E 28		LD	A, 40	;160-cycle wait
	3D	L1:	DEC	A	
	20 FD		JR	NZ, L1	
	C9		RET		

2. Example of program that writes the above instructions to internal RAM starting from 0xFF80:

```
            Label       Instruction
            LD          C, 80H
            LD          B, 10
            LD          HL, DMADATA
L2:         LD          A, (HLI)
            LD          (C), A
            INC         C
            DEC         B
            JR          NZ, L2
            •
            •
            •
DMADATA     DB      3EH,    0C0H,   0E0H,   46H,    3EH
            DB      28H,    3DH,    20H,    0FDH,   0C9H
```

3) When the DMA transfer is performed, the subroutine written to internal RAM shown in 1) above is executed:

```
.
CALL   0FF80h          ;DMA transfer
.
```

Note: ***The preceding program is used for DMA transfers performed within routines for processing interrupts implemented by vertical blanking. In all other cases, however, the program written to internal RAM should be as shown below to prevent interrupts during a transfer.***

Address	Command	Label	Instruction		Comment
FF80	F3		DI		;Interrupt disabled
	3E C0		LD	A, 0C0H	
	E0 46		LD	(DMA), A	;C000-C09F→OAM
	3E 28		LD	A, 40	;160-cycle wait
	3D	L1:	DEC	A	
	20 FD		JR	NZ, L1	
	FB		EI		;Interrupt enabled
	C9		RET		

Example 2

The example below shows a DMA transfer of 40 x 32 bits of data from the display RAM area (9F00h-9F9Fh) to OAM (FE00-FE9Hh).

Machine Code	Label	Instruction		Comment
3E 9F		LD	A, 9FH	
E0 46		LD	(DMA), A	;9F00-9F9F→OAM

Data can be transferred by DMA from 0x8000-0x9F9F to OAM either by the method shown in Example 1 or by using only the above instructions.

2.1.8.2 DMA Transfers in CGB

Using the Earlier DMA Transfer Method

This DMA method transfers only 40 x 32 bits of data from 0-DFFFh to OAM (FE00h-FE9Fh). The transfer starting address can be specified as 0-DFFFh in increments of 100h. The transfer method is the same as that used in DMG, but when data are transferred from 8000h-9FFFh (LCD display RAM area), the data transferred are those in the bank specified by bit 0 of register VBK. When transferring data from D000h-DFFFh (unit working RAM area), the data transferred are those in the bank specified by the lower 3 bits of register SVBK.

Note: When the CPU is operating at double-speed, the transfer rate is also doubled.

Using the New DMA Transfer Method

The DMA transfer method provided for DMG has been augmented in CGB with the following DMA transfer functions.

1. Horizontal Blanking DMA Transfer

Sixteen bytes of data are automatically transferred from the user program area (0-7FFFh) and external and unit working RAM area (A000h-DFFFh) to the LCD display RAM area (8000h-9FFFh) during each horizontal blanking period. The number of lines transferred by DMA in a horizontal blanking period can be specified as 1-128 by setting register HDMA5. CPU processing is halted during a DMA transfer period.

2. General-Purpose DMA Transfers

Between 16 and 2048 bytes (specified in 16-byte increments) are transferred from the user program area (0-7FFFh) and external and unit working RAM area (A000h-DFFFh) to the LCD display RAM area (8000h-9FFFh). As with horizontal blanking DMA transfers, CPU operation is halted during the DMA transfer period.

The unit working RAM area (D000h-DFFFh) selected as the transfer source is the bank specified by register SVBK.

The LCD display RAM area (8000h-9FFFh) selected as the transfer destination is the bank specified by register VBK.

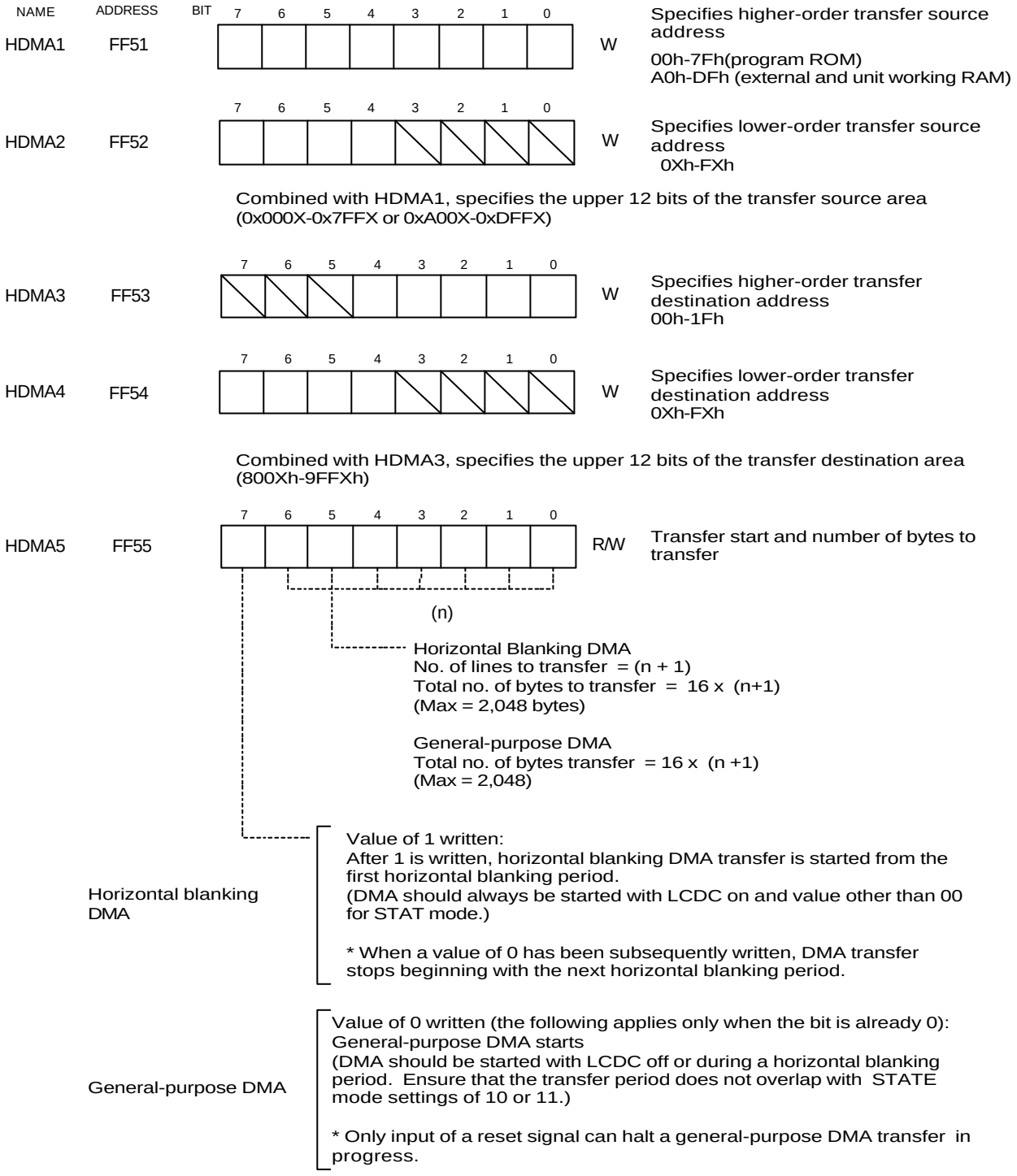
Special Notes

- The number of bytes transferred by the new DMA method must be specified in 16-byte increments; byte counts that are not a multiple of 16 cannot be transferred.
- With the new DMA transfer method, transfers are performed at a fixed rate regardless of whether the CPU is set to operate at normal or double-speed.
- Horizontal blanking DMAs should always be started with the LCDC on and the STAT mode set to a value other than 00.
- General-purpose DMAs should be performed with the LCDC off or during a vertical blanking period.
- When the new DMA transfer method is used to transfer data from the user program area (0-7FFFh), mask ROM and MBC for double-speed mode are required.

2.1.8.3 DMA Control Register: DMG and CGB

NAME	ADDRESS	BIT 7	6	5	4	3	2	1	0		
DMA	FF46h									W	DMA Transfer and Starting Address

2.1.8.4 New DMA Control Registers: CGB only

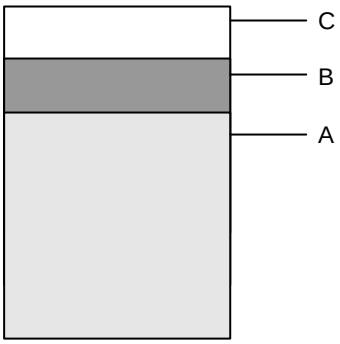

NAME
ADDRESS
BIT
7 6 5 4 3 2 1 0
HDMA1
FF51
W
Specifies higher-order transfer source address
00h-7Fh(program ROM)
A0h-DFh (external and unit working RAM)
HDMA2
FF52
W
Specifies lower-order transfer source address
0Xh-FXh
Combined with HDMA1, specifies the upper 12 bits of the transfer source area
(0x000X-0x7FFX or 0xA00X-0xDFFX)
HDMA3
FF53
W
Specifies higher-order transfer destination address
00h-1Fh
HDMA4
FF54
W
Specifies lower-order transfer destination address
0Xh-FXh
Combined with HDMA3, specifies the upper 12 bits of the transfer destination area
(800Xh-9FFXh)
HDMA5
FF55
R/W
Transfer start and number of bytes to transfer
(n)
Horizontal Blanking DMA
No. of lines to transfer = (n + 1)
Total no. of bytes to transfer = 16 x (n+1)
(Max = 2,048 bytes)
General-purpose DMA
Total no. of bytes transfer = 16 x (n +1)
(Max = 2,048)
Horizontal blanking DMA
Value of 1 written:
After 1 is written, horizontal blanking DMA transfer is started from the first horizontal blanking period.
(DMA should always be started with LCDC on and value other than 00 for STAT mode.)
* When a value of 0 has been subsequently written, DMA transfer stops beginning with the next horizontal blanking period.
General-purpose DMA
Value of 0 written (the following applies only when the bit is already 0):
General-purpose DMA starts
(DMA should be started with LCDC off or during a horizontal blanking period. Ensure that the transfer period does not overlap with STATE mode settings of 10 or 11.)
* Only input of a reset signal can halt a general-purpose DMA transfer in progress.

Note: These registers cannot be written to in DMG mode.

2.1.9 OBJ Display Priority

As a rule, when objects overlap, the one with the lower OBJ number is given priority. In DMG or CGB in DMG mode, among overlapping objects with different x-coordinates, priority is given to the object with the smallest x-coordinate.

1. Same x-coordinate: For both DMG and CGB



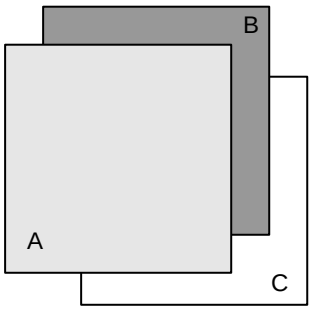
a = No. of OBJ A
b = No. of OBJ B
c = No. of OBJ C

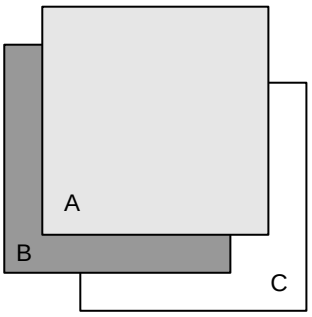
When a < b < c, objects are displayed as indicated in the figure at left.

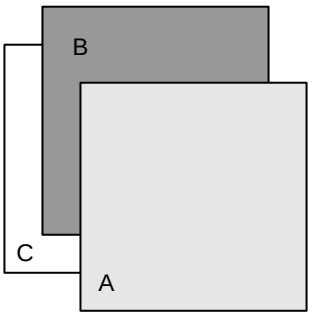
2. Different x-coordinates: CGB Only

a = No. of OBJ A
b = No. of OBJ B
c = No. of OBJ C

When a < b < c, objects are is displayed as indicated in the figures below.

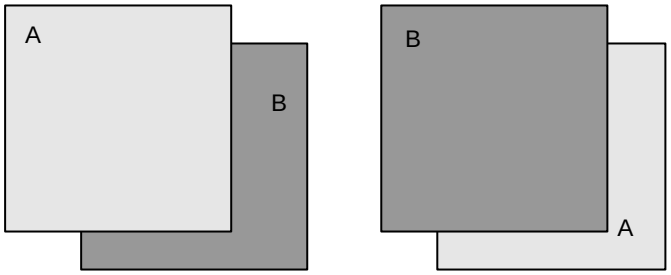






3. Different x-coordinates: DMG/CGB in DMG Mode

In DMG mode and with objects with different x-coordinates, the object with the smallest x-coordinate is given priority.



a = No. of OBJ A
b = No. of OBJ B

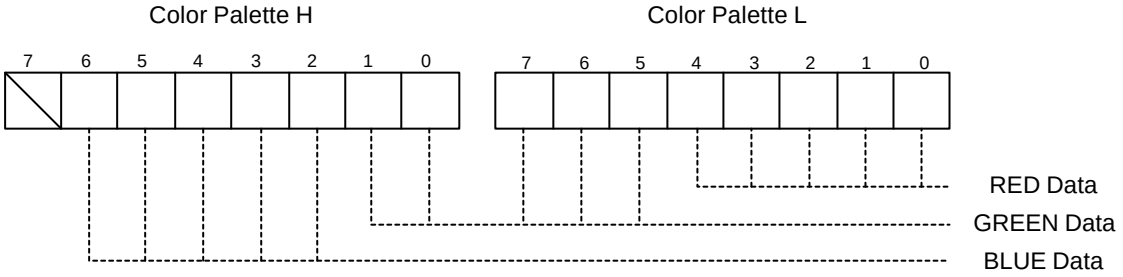
When a < b, objects are displayed as indicated in the figure at left.

2.2 LCD COLOR DISPLAY (CGB ONLY)

The LCD unit of the CGB system can display 32 shades each for RGB, for a total 32,768 colors. A single color palette consists of 4 colors selected from among these 32,768. One of 8 palettes can be selected for each BG and OBJ character. However, because each OBJ includes transparent data, each OBJ color palette consists of 3 colors. The color palettes for BG and OBJ are independent of one another.

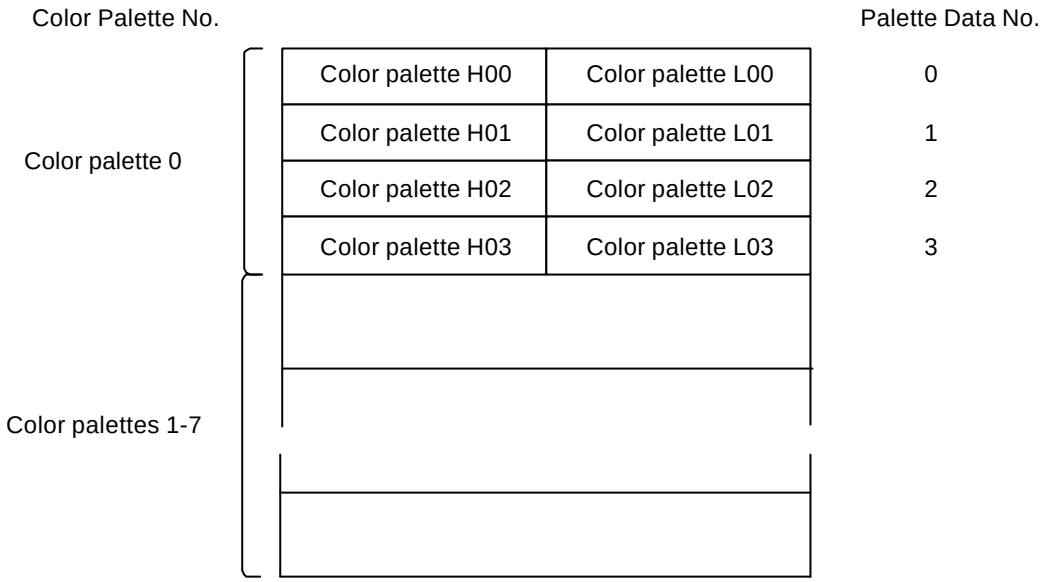
2.2.1 Color Palettes

- Eight palettes each are provided for BG and OBJ.
- Each palette consists of 4 colors and is specified by the display dot data (2 bits) (palette data nos. 0-3).
- The color palettes represent each color with 2 bytes, with 5 bits of data for each color of RGB (32,768 displayable colors).



2.2.2 Color Palette Composition

1. BG Color Palettes



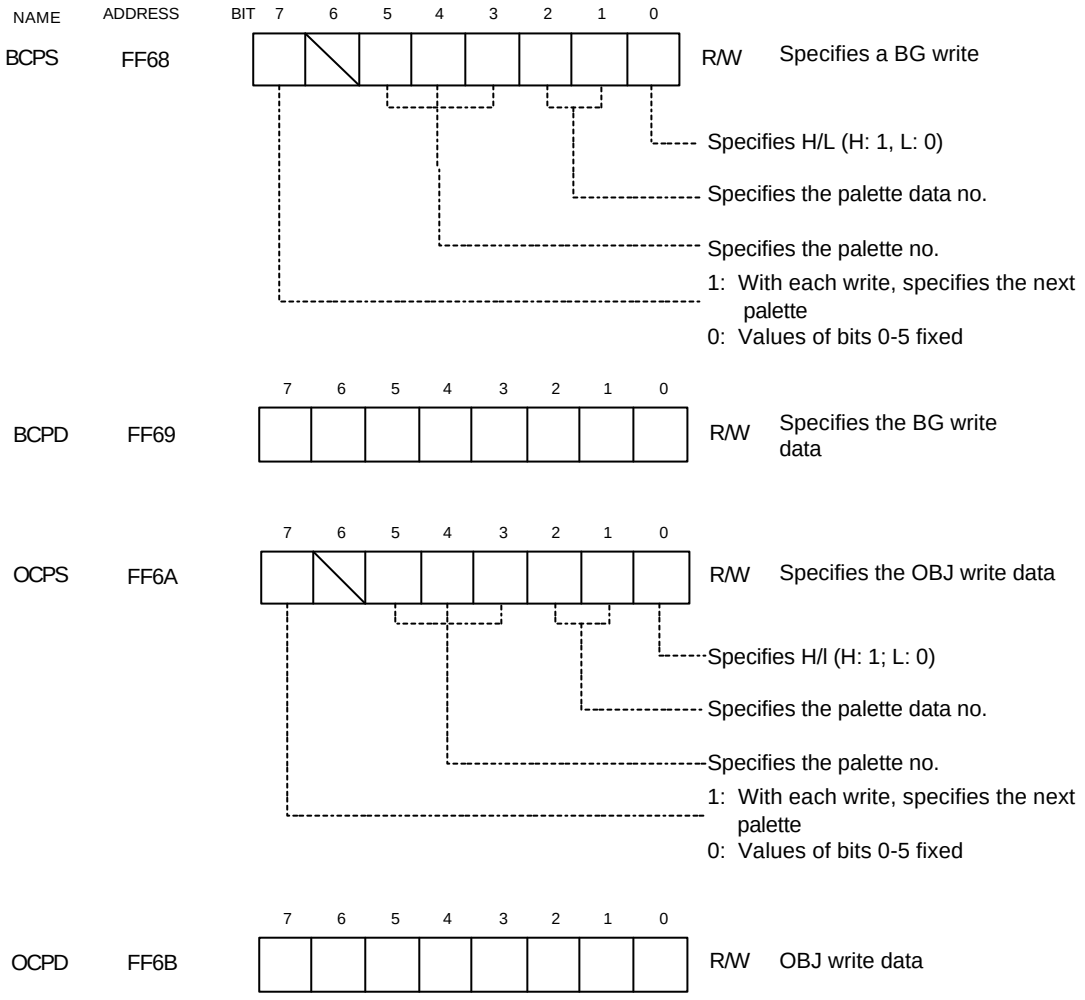
2. OBJ Color Palettes

OBJ color palettes have the same composition as shown in the previous figure.

2.2.3 Writing Data to a Color Palette

Data are written to color palettes using the write-specification and write-data registers. The lower 6 bits of the write-specification register specifies the write address. Data are written to the write-data register, at the address specified by the write-specification register. If the most significant bit of the write-specification register is set to 1, the write address is then automatically incremented to specify the next address. (The next address is read from the lower 6 bits of the write-specification register.)

The write-specification and write-data registers also are used to read data from color palettes. Data are read from the write-data register, and the read data are those at the address specified by the write-specification register. When data are read, the specified address is not incremented even if the most-significant bit of the write-specification register is set to 1.



Note: These registers cannot be written to in DMG mode.

2.2.4 Overlapping OBJ and BG

When objects are displayed, overlapping objects and background are displayed according to the display priority flags for OBJ and BG, as indicated below. The BG display priority flag can be used to assign BG display priority to individual characters.

Display Priority Flag		Dot Data		Screen Display	
BG	**OBJ**	**OBJ**	**BG**	**Palette**	**Data**
0: Use OBJ Priority	0: Priority to OBJ	00 00 obj obj	00 bg 00 bg	BG BG OBJ OBJ	00 bg obj obj
	1: Priority to BG	00 00 obj obj	00 bg 00 bg	BG BG OBJ BG	00 bg obj bg
1: Highest Priority to BG (by character)	0 1	00 00 obj obj	00 bg 00 bg	BG BG OBJ BG	00 bg obj bg

Note: obj and bg represent dot data (01, 10, 11) for OBJ and BG, respectively.

2.2.5 Display Using Earlier DMG Software (DMG mode)

When earlier DMG software is used, coloring is performed automatically by the system using registers BGP, 0BP0, and 0BP1. However, the display uses 3 palettes, 1 for BG, with 4 colors, and 2 for OBJ, each with 3 colors (excluding transparent; maximum of 10 colors in 1 screen).

1. BG Display

   Colors specified in BG color palette No. 0 are displayed by the dot data (2 bits) whose grayscales are specified by register BGP.

2. OBJ Display

   Colors specified in OBJ color palettes No. 0 and No. 1 are displayed by the dot data (2 bits) whose grayscales are specified by registers OBP0 and OBP1.

   The CGB hardware automatically selects the display color according to the color palette pre-registered in the CGB (cannot be changed by a program). However, when turning on power to the CGB, the player can select from a combination of the 12 colors registered in the unit. This function is available only in DMG mode.

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 3 SOUND FUNCTIONS 81

3.1 Overview of Sound Functions 81

3.2 Sound Control Registers 83

3.2.1 Sound 1 Mode Registers 83
3.2.2 Sound 2 Mode Registers 86
3.2.3 Sound 3 Mode Registers 88
3.2.4 Sound 4 Mode Registers 90
3.2.5 Sound Control Registers 92

3.3 VIN Terminal Usage Notes 94

Revision History

Date	Section	Description
12/3/99	3.1	Addition of "Usage Notes"
12/3/99	3.2.1 3.2.2 3.2.4	Revision and addition of Notes in Mode Registers for Sound 1, 2, 4
4/25/00	3.2.1 3.2.2 3.2.3	Addition of Notes in Mode Registers for Sound 1, 2, 3
10/10/00	3.2.1 3.2.2 3.2.3 3.2.4 3.2.1 3.2.2	Added Cautions for Sound 1, 2, 3, 4 Revised Cautions for Sound 1 Revised drawing for NR23 and NR24

CHAPTER 3: SOUND FUNCTIONS

3.1 Overview of Sound Functions

The sound circuitry consists of circuits that generate 4 types of sounds (Sounds 1-4). It can also synthesize external audio input waveforms and output sounds. (External audio input is a function available only in CGB).

Sound 1: Generates a rectangle waveform with sweep and envelope functions.
Sound 2: Generates a rectangle waveform with an envelope function.
Sound 3: Outputs any waveform from waveform RAM.
Sound 4: Generates white noise with an envelope function.

Each sound has two modes, ON and OFF.

- ON Mode
  Sounds are output according to data in the mode register for each sound.
  The mode register data can be specified as needed while outputting sound.

  - Initialization Flag
    When the default envelope values are set and the length counter is restarted, the initialization flag is set to 1 and the data is initialized.

  - Mute
    In the following instances, the synthesizer will enter mute status. No sound will be output regardless of the ON flag setting.

    Sounds 1, 2, and 4:
    -When the output level is 0 with the default envelope value set to a value other than 0000 and in DOWN mode
    -When the step is 0 with the default envelope value set to a value of 0000 and in UP mode (NR12, NR22, and NR42 set to 0x08h and the initialization flag set)

    Sound 3:
    With the output level set to mute
    (bits 5 and 6 of NR32 set to 0)

  - Stop Status
    In the following cases, the ON flag is reset and sound output is halted.
    -Sound output is halted by the length counter.
    -With Sound 1, during a sweep operation, an overflow occurs in addition mode.

♦ OFF Mode
Stops operation of the frequency counter and D/A converter and halts sound output.

♦ Sounds 1, 2, and 4:
-When the default level is set to 0000 with the envelope in DOWN mode (initialization not required)

♦ Sound 3:
-When the Sound OFF flag (bit 7 of NR30) is set to 0.
Setting the Sound OFF flag to 1 cancels OFF mode.
Sound 3 is started by re-initialization.

♦ All Sounds OFF mode
-Setting the All Sounds ON/OFF flag (bit 7 of NR52) to 0 resets all of the mode registers (for sounds 1, 2, 3, and 4) and halts sound output. Setting the All Sounds ON/OFF flag to 1 cancels All Sounds OFF mode.

Note: The sound mode registers should always be set after All Sound OFF mode is cancelled. The sound mode registers cannot be set in All Sound OFF mode.

♦ Sound Usage Notes
Use one of the following methods to halt sounds 1, 2, or 4.

1) Use NR51.
2) Set NR12, NR22, and NR42 to 0x08h.
3) Set NR14, NR24, and NR44 to 0x80h.

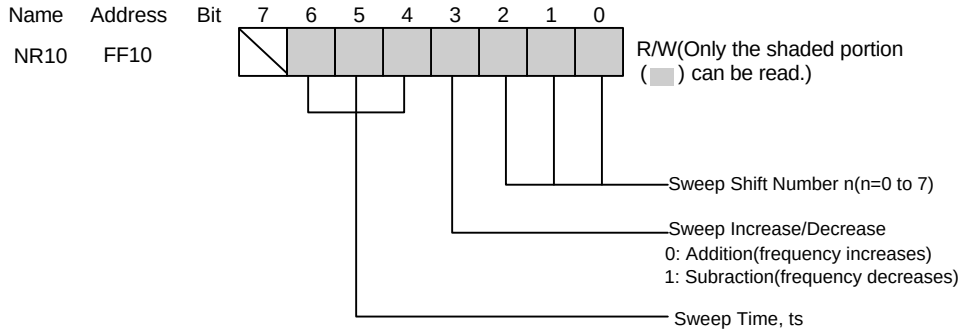
Switch to OFF mode during the scene you stop the BGM. Unless you switch to OFF mode, a faint whining noise can be heard due to the sound circuit structure.

3.2 Sound Control Registers

3.2.1 Sound 1 Mode Registers

Sound 1 is a circuit that generates a rectangle waveform with sweep and envelope functions.
It is set by registers NR10, NR11, NR12, NR13, and NR14.

♦ Sweep Shift Number



♦ Sweep Shift Number

The frequency with one shift (NR13 and NR14) is determined by the following formula.

$X(t) = X(t-1) \pm X(t-1) / 2^n$ n = 0 to 7

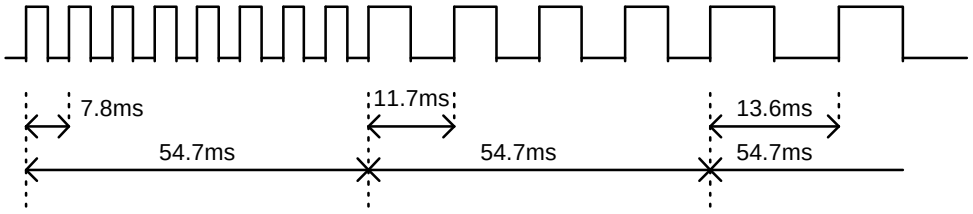
X (0) = default data X (t-1) is the previous output frequency

If the result of this formula is a value consisting of more than 11 bits, sound output is stopped and the Sound 1 ON flag of NR52 (bit 0) is reset.
In a subtraction operation, if the subtrahend is less than 0, the result is the pre-calculation value X (t) = X (t -1). However, if n = 0, shifting does not occur and the frequency is unchanged.

♦ Sweep time (ts)

Frequency varies with each value of ts.
000: Sweep OFF
001: ts=1/f128 (7.8ms)
010: ts=2/f128 (15.6ms)
011: ts=3/f128 (23.4ms)
100: ts=4/f128 (31.3ms)
101: ts=5/f128 (39.1ms)
110: ts=6/f128 (46.9ms)
111: ts=7/f128 (54.7ms) f128=128Hz

Example: When NR10 = 79h and the default frequency = 400h, the sweep waveform appears as follows.



Note: When the sweep function is not used, the increase/decrease flag should be set to 1 (subtraction mode).

Name	Address	Bit 7	6	5	4	3	2	1	0
NR11	FF11								

R/W(Only the shaded portion ( ) can be read.)

Sound Length, t1(0 to 63)

Waveform Duty Cycle

Sound length = (64 - t1) x (1/256) sec

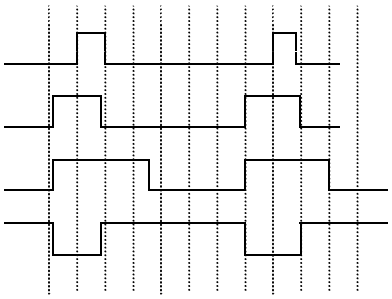
Waveform Duty Cycles

00 : 12.5%

01 : 25%

10 : 50%

11 : 75%



Name	Address	Bit 7	6	5	4	3	2	1	0
NR12	FF12								

R/W

Length of Envelope Steps n(n=0 to 7)

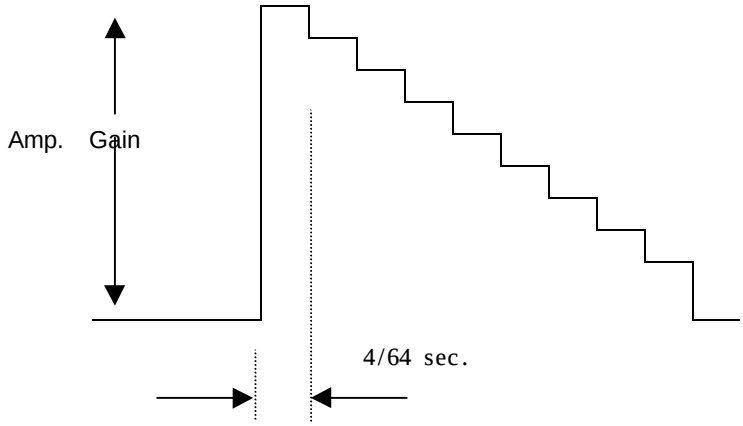
Envelope Up/Down
0: Attenuate(decrease)
1: Amplify(increase)

Default Envelope Value

Length of Envelope Steps:
Sets the length of each step of envelope amplification or attenuation.
Length of 1 step = N x (1/64) sec
When N = 0, the envelope function is stopped.

Default Envelope Value (0000 to 1111_B):
16 step levels can be specified using the 4-bit D/A circuit.
Maximum is 1111_B, and 0000 is the mute setting.

Example: When NR12 = 94h, the Amp Gain is as follows.



Note: By Setting the envelope register only nothing will be reflected in the output. Always set the initial flag.

Name	Address	Bit	7	6	5	4	3	2	1	0	
NR13	FF13										W (Low-order Frequency Data)

Name	Address	Bit	7	6	5	4	3	2	1	0	
NR14	FF14										R/W(Only the shaded portion ( ) can be read.)

High-order Frequency Data(3 bits)

Counter Continuous Selection

Initialize

Counter/Continuous Selection
0: Outputs continuous sound regardless of length data in register NR11.
1: Outputs sound for the duration specified by the length data in register NR11. When sound output is finished, bit 0 of register NR52, the Sound 1 ON flag, is reset.

Initialize
Setting this bit to 1 restarts Sound 1.
With the 11-bit frequency data specified in NR13 and NR14 represented by x, the frequency, f, is determined by the following formula.
$f = 4194304 / (4 \times 2^3 \times (2048 - X))$ Hz
Thus, the minimum frequency is 64 Hz and the maximum is 131.1 KHz.

♦ Sound 1 Usage Notes

-When the sweep function is not used, set the sweep time to 0 (sweep OFF) and the sweep increase/decrease flag to 1.

-Sound may not be produced if the sweep increase/decrease flag of NR10 is set to 0 (addition mode), the sweep shift number set to a value other than 0, and the mode set to sweep OFF (e.g. NR10 = 01h).

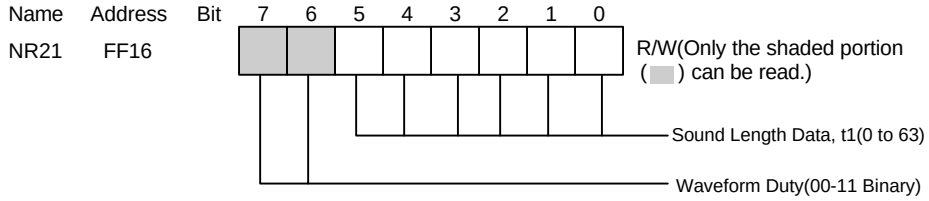
-When a value is written in the envelope register, the sound output becomes unstable till the initial flag is set. Therefore, set the initial flag immediately after writing a value in the envelope register.

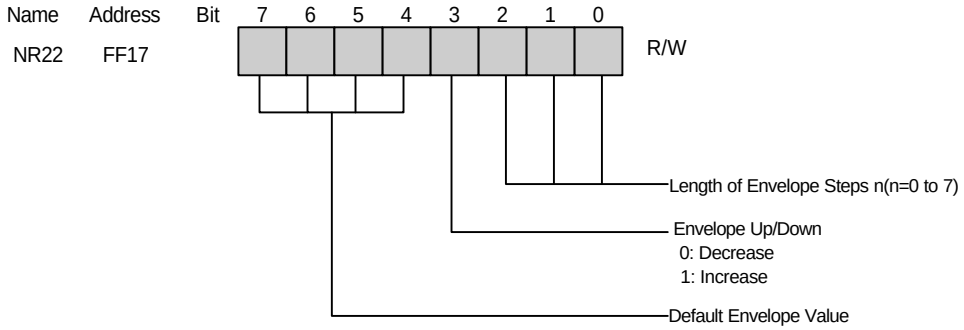
-If High-order Frequency Data (lower 3 bits of NR14) is changed when consecutive operation mode (set bit 6 of NR14 to 0) is selected with Sound 1, sound length data (bits 0-5 of NR11) should be set to 0 after frequency data is set. Otherwise, sound may stop prematurely.

-Do not play back audio (speech or song) that uses Sound 1, 2, and 4. It cannot be played back normally with AGB's CGB compatible mode.

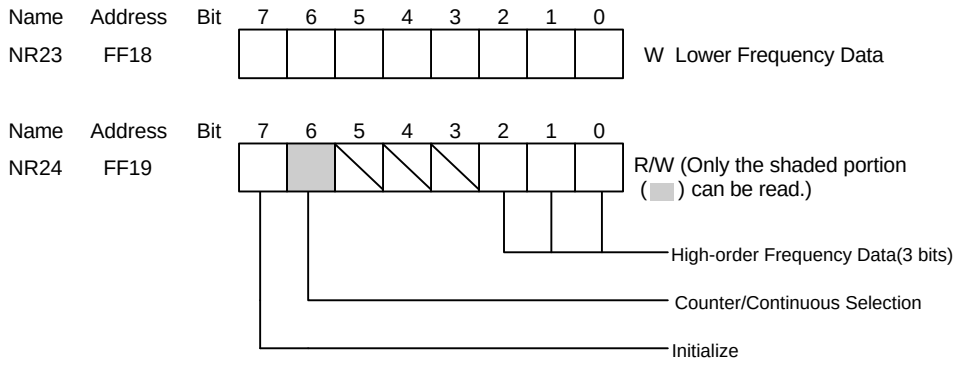
3.2.2 Sound 2 Mode Registers

Sound 2 is a circuit that generates a rectangle waveform with an envelope function. It is set by registers NR21, NR22, NR23, and NR24.





Note: By Setting the envelope register only, nothing will be reflected in the output. Always set the initial flag.



Counter/Continous Selection
0: Outputs continuous sound regardless of length data in register NR21.
1: Outputs sound for the duration specified by the length data in register NR21. When sound output is finished, bit 1 of register NR52, the Sound 2 ON flag, is reset.

Initialize
Setting this bit to 1 restarts Sound 2.

♦ Sound 2 Usage Notes

When a value is written in the envelope register, the sound output becomes unstable until the initial flag is set. Hence, set the initial flag immediately after writing a value in the envelope register.

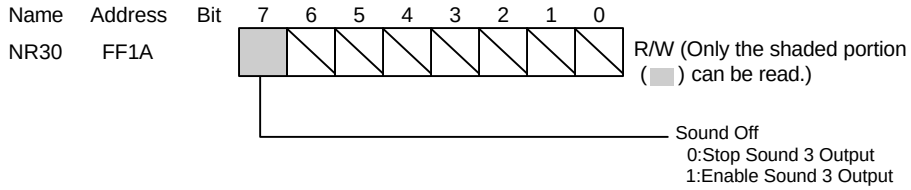
If High-order Frequency Data (lower 3 bits of NR24) is changed when consecutive operation mode (set bit 6 of NR24 to 0) is selected with Sound 2, sound length data (bits 0-5 of NR21) should be set to 0 after frequency data is set. Otherwise, sound may stop prematurely.

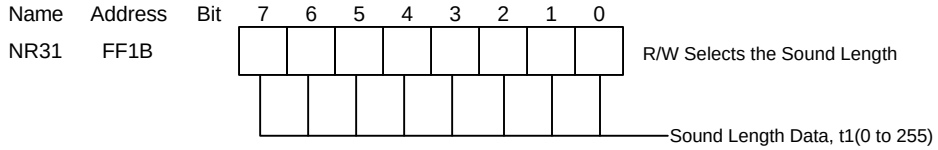
Do not play back audio (speech or song) that uses Sound 1, 2, and 4. It cannot be played back normally with AGB's CGB compatible mode.

3.2.3 Sound 3 Mode Registers

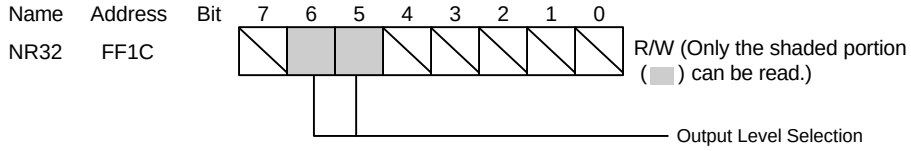
Sound 3 is a circuit that generates user-defined waveforms. It automatically reads a waveform pattern (1 cycle)written to waveform RAM at FF30h-FF3Fh, and it can output a sound while changing its length, frequency, and level by registers NR30, NR31, NR32, NR33, and NR34.

The settings of the sound length and frequency functions and data are the same as for the Sound 1 circuit.

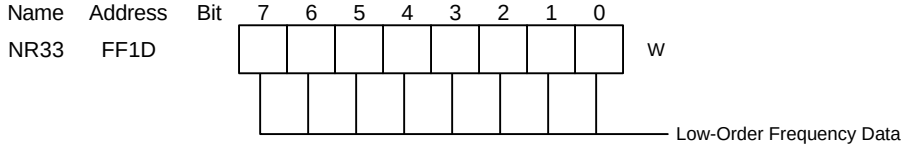

Name Address Bit 7 6 5 4 3 2 1 0
NR30 FF1A
R/W (Only the shaded portion
( ) can be read.)
Sound Off
0:Stop Sound 3 Output
1:Enable Sound 3 Output

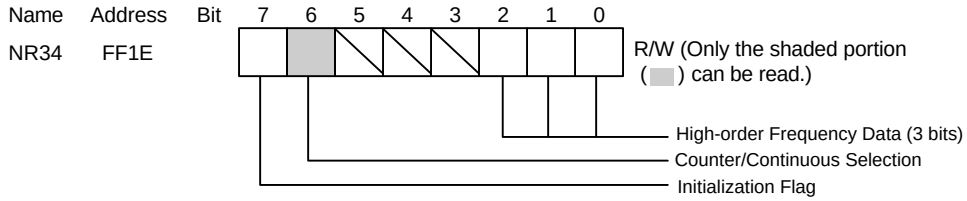


Sound Length = (256-t1) x (1/256) sec



Output Level:
00: Mute
01: Output waveform RAM data (4-bit length) unmodified.
10: Output waveform RAM data (4-bit length) shifted 1 bit to the right (1/2).
11: Output waveform RAM data (4-bit length) shifted 2 bits to the right (1/4).





Counter/Continous Selection
0: Outputs continuous sound regardless of length data in register NR31.
1: Outputs sound for the duration specified by the length data in register NR31. When sound output is finished, bit 2 of register NR52, the Sound 3 ON flag, is reset.

Initialization Flag
When the Sound OFF flag (bit 7, NR30) is set to 1, setting this bit to 1 restarts Sound 3.

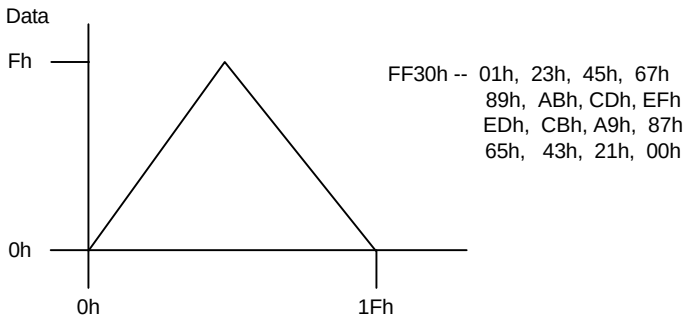
♦ Sound 3 Usage Notes

- The initialization flag should not be set when the frequency is changed during Sound 3 output. Setting the initialization flag during Sound 3 operation (Sound 3 ON flag = 1) may destroy the contents of waveform RAM.
- Setting the initialization flags for Sound 1, Sound 2, or Sound 4 does not cause a problem.

♦ Waveform RAM Composition
Waveform RAM consists of waveform patterns of 4 bits x 32 steps.

Address	D7	D6	D5	D4	D3	D2	D1	D0
FF30		Step 0				Step 1		
FF31		Step 2				Step 3		
FF32		Step 4				Step 5		
⋮								
FF3F		Step 30				Step 31		

Example: Triangular Wave



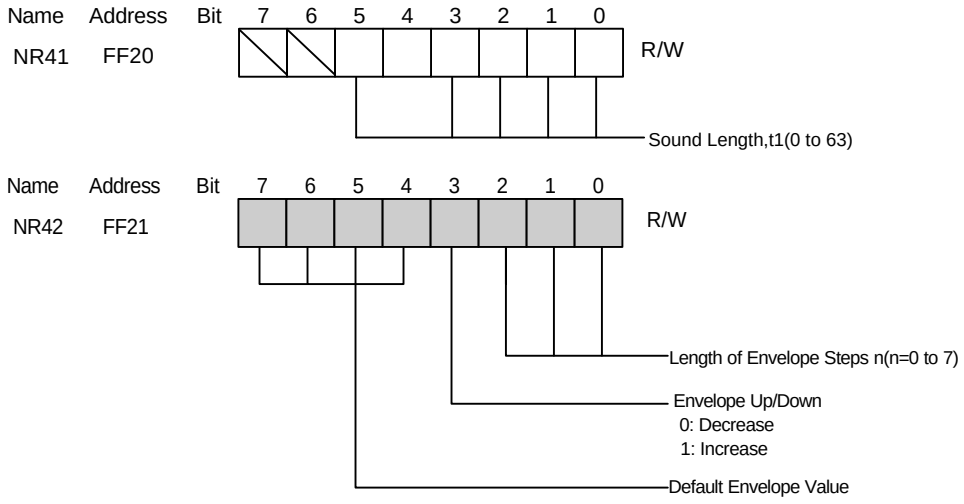
The initialization flag should not be set when the frequency is changed during Sound 3 output.

If High-order Frequency Data (lower 3 bits of NR34) is changed when consecutive operation mode (set bit 6 of NR34 to 0) is selected with Sound 3, sound length data (bits 0-7 of NR31) should be set to 0 after frequency data is set. Otherwise, sound may stop prematurely.

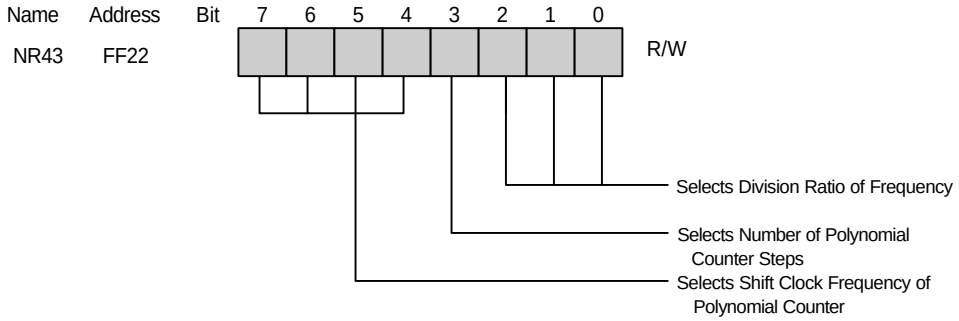
When you play back audio, by updating the Sound 3 Waveform RAM regularly, noise may occur with AGB's CGB compatible mode. Always confirm the sound output with Game Boy Advance. For example, if Sound 3 is turned ON and OFF (bit 7 of NR30) repeatedly, it could cause noise.

3.2.4 Sound 4 Mode Registers

Sound 4 is a white-noise generating circuit. It can output sound while switching the number of steps of the polynomial counter for random number generation and changing the frequency dividing ratio and envelope data by registers NR41, NR42, NR43, and NR44.


Name Address Bit 7 6 5 4 3 2 1 0
NR41 FF20
R/W
Sound Length,t1(0 to 63)
Name Address Bit 7 6 5 4 3 2 1 0
NR42 FF21
R/W
Length of Envelope Steps n(n=0 to 7)
Envelope Up/Down
0: Decrease
1: Increase
Default Envelope Value

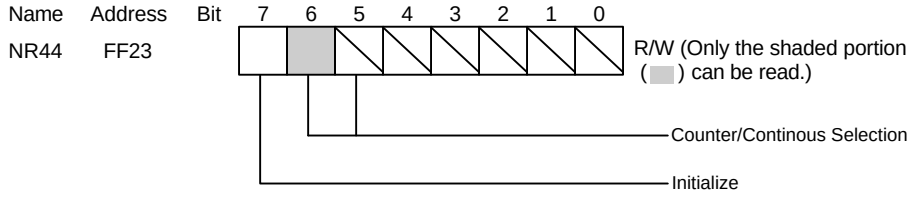
Note: By only setting the envelope register nothing will be reflected in the output. Always set the initial flag.



Selecting the dividing ratio of the frequency:
Selects a 14-step prescalar input clock to produce the shift clock for the polynomial counter.
000 : $fx1/2^3x2$
001 : $fx1/2^3x1$
010 : $fx1/2^3x1/2$
011 : $fx1/2^3x1/3$
100 : $fx1/2^3x1/4$
101 : $fx1/2^3x1/5$
110 : $fx1/2^3x1/6$
111 : $fx1/2^3x1/7$ f=4.194304MHz

Selecting the number of steps for the polynomial counter:
0: 15 steps
1: 7 steps

Selecting the shift clock frequency of the polynomial counter:
0000: Dividing ratio frequency x $1/2$
0001: Dividing ratio frequency x $1/2^2$
0010: Dividing ratio frequency x $1/2^3$
0011: Dividing ratio frequency x $1/2^4$
: :
: :
1101: Dividing ratio frequency x $1/2^{14}$
1110: Prohibited code
1111: Prohibited code



Counter/Continuous Selection:
0: Outputs continuous sound regardless of length data in register NR41.
1: Outputs sound for the duration specified by the length data in register NR41. When sound output is finished, bit 3 of register NR52, the Sound 4 ON flag, is reset.

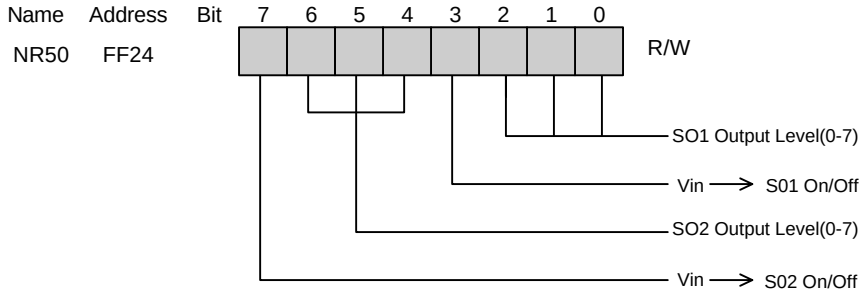
Initialize:
Setting this bit to 1 restarts Sound 4.

♦ Sound 4 Usage Notes

When a value is written in the envelope register, the sound output becomes unstable until the initial flag is set. Hence, set the initial flag immediately after writing a value in the envelope register.

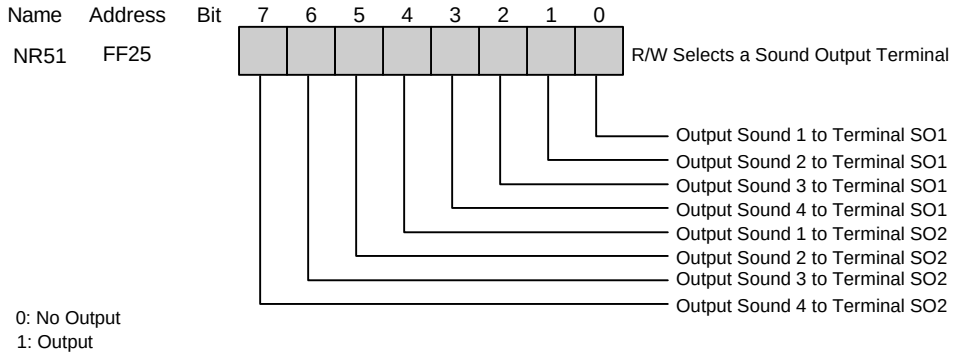
Do not play back audio (speech or song) that uses Sound 1, 2, and 4. It cannot be played back normally with AGB's CGB compatible mode.

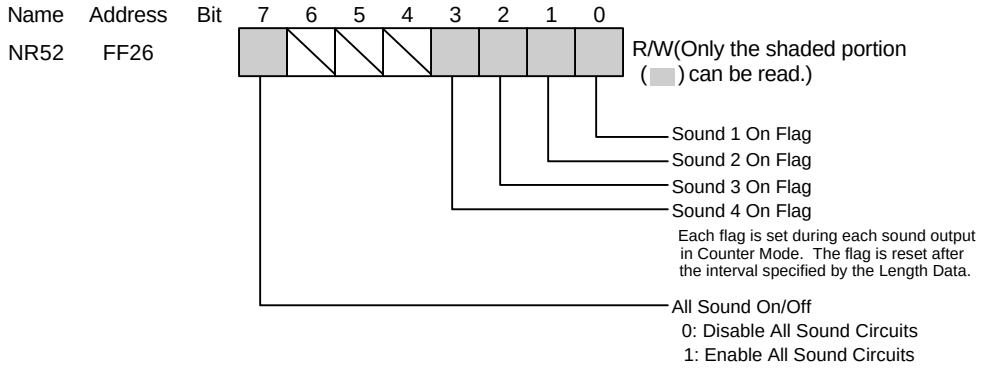
3.2.5 Sound Control Registers



Output Level:
000: Minimum level (Maximum level÷8)
:
:
111: Maximum level

V i n→SO1 ON/OFF (V i nSO2→ON/OFF)
Synthesizes audio input from Vin terminal with sounds 1-4 and ouputs the result.
0: No output
1: Output


Name
Address
Bit
7 6 5 4 3 2 1 0
NR51
FF25
R/W Selects a Sound Output Terminal
Output Sound 1 to Terminal SO1
Output Sound 2 to Terminal SO1
Output Sound 3 to Terminal SO1
Output Sound 4 to Terminal SO1
Output Sound 1 to Terminal SO2
Output Sound 2 to Terminal SO2
Output Sound 3 to Terminal SO2
Output Sound 4 to Terminal SO2
0: No Output
1: Output


Name
Address
Bit
7 6 5 4 3 2 1 0
NR52
FF26
R/W(Only the shaded portion
( ) can be read.)
Sound 1 On Flag
Sound 2 On Flag
Sound 3 On Flag
Sound 4 On Flag
Each flag is set during each sound output
in Counter Mode. The flag is reset after
the interval specified by the Length Data.
All Sound On/Off
0: Disable All Sound Circuits
1: Enable All Sound Circuits

3.3 VIN Terminal Usage Notes

- The VIN terminal can be used normally only in CGB. (Since the signal from the VIN terminal is too low to be used, the VIN terminal cannot be used in DMG.)

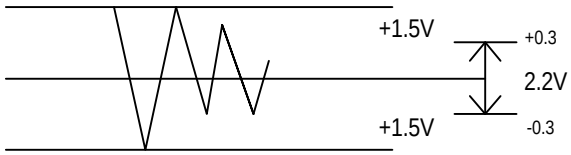
- The maximum amplitude of the synthesized output is 3V.

- The design prevents the maximum amplitude from exceeding 3V when only sounds 1-4 are used, even when the output level for each sound is set to the maximum.

When the output level is set to 0Fh, each sound is output at 0.75V.
0.75V x 4 = 3V

- The maximum amplitude of the synthesized sound output also must be limited to 3V or less when the VIN terminal is used to input external sound.

Example: Using Sounds 1-4 and the VIN terminal
Use software to adjust the output levels of sounds 1-4 so that they do not exceed 0.6V (3V ÷ 5 = 0.6V).Also limit the output level of the VIN terminal to 0.6V or less (input range of 1.9 - 2.5V).



- The input voltage from the VIN terminal also can be increased if the levels of the internal sounds are low or if not all 4 sounds are used (total output level of 3V or less).

CHAPTER 4: CPU INSTRUCTION SET 97

4.1 General Purpose Registers 97

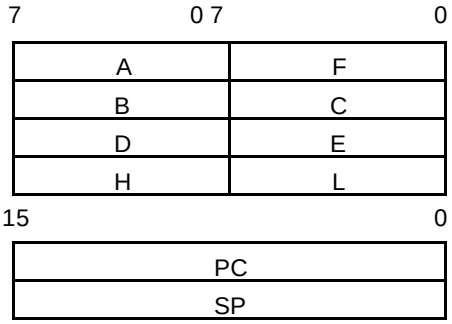
4.2 Description of Instructions 98

4.2.1 8-Bit Transfer and Input/Output Instructions.....98
4.2.2 16-Bit Transfer Instructions.....103
4.2.3 8-Bit Arithmetic and Logical Operation Instructions.....105
4.2.4 16-Bit Arithmetic Operation Instructions..... 111
4.2.5 Rotate Shift Instructions.....113
4.2.6 Bit Operations.....118
4.2.7 Jump Instructions.....120
4.2.8 Call and Return Instructions.....122
4.2.9 General-Purpose Arithmetic Operations and CPU Control Instructions.....125

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 4: CPU INSTRUCTION SET

4.1 General Purpose Registers

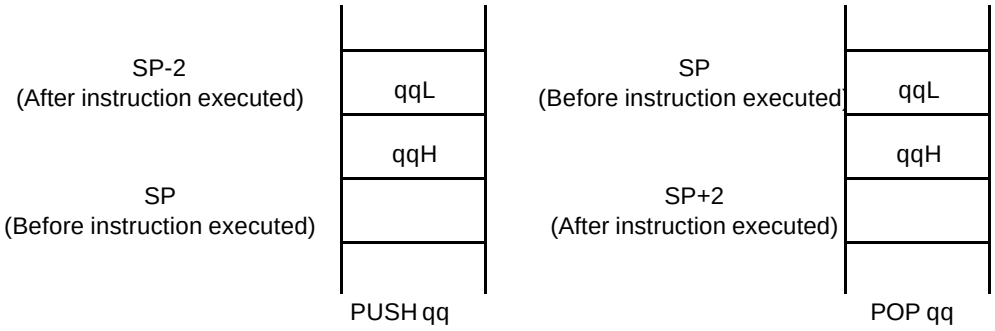


- Accumulator: A
  An 8-bit register for storing data and the results of arithmetic and logical operations.

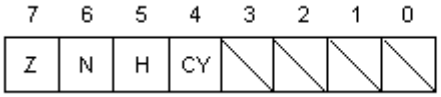
- Auxiliary registers: B, C, D, E, F, H, and L
  These serve as auxiliary registers to the accumulator. As register pairs (BC, DE, HL), they are 8-bit registers that function as data pointers.

- Program counter: PC
  A 16-bit register that holds the address data of the program to be executed next.
  Usually incremented automatically according to the byte count of the fetched instructions. When an instruction with branching is executed, however, immediate data and register contents are loaded.

- Stack pointer: SP
  A 16-bit register that holds the starting address of the stack area of memory.
  The contents of the stack pointer are decremented when a subroutine CALL instruction or PUSH instruction is executed or when an interrupt occurs and incremented when a return instruction or pop instruction is executed.



- Flag Register: F
  Consists of 4 flags that are set and reset according to the results of instruction execution.
  Flags CY and Z are tested by various conditional branch instructions.



  Z: Set to 1 when the result of an operation is 0; otherwise reset.
  N: Set to 1 following execution of the substruction instruction, regardless of the result.
  H: Set to 1 when an operation results in carrying from or borrowing to bit 3.
  CY: Set to 1 when an operation results in carrying from or borrowing to bit 7.

4.2 Description of Instructions

4.2.1 8-Bit Transfer and Input/Output Instructions

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD r, r'	r ← r'	--	--	--	--	1	0 1 r r'

Loads the contents of register r' into register r.

Codes for registers r and r'

Register	r, r'
A	111
B	000
C	001
D	010
E	011
H	100
L	101

Examples: LD A, B ; A ← B
LD B, D ; B ← D

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD r, n	r ← n	--	--	--	--	2	00 r 110
							← n →

Loads 8-bit immediate data n into register r.

Example: L D B, 24h ; B ← 24h

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD r, (HL)	r ← (HL)	--	--	--	--	2	01 r 110

Loads the contents of memory (8 bits) specified by register pair HL into register r.

Example: When (HL) = 5Ch,
LD H, (HL) ; H ← 5Ch

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD (HL), r	(HL) ← r	--	--	--	--	2	01 110 r

Stores the contents of register r in memory specified by register pair HL.

Example: When A = 3Ch, HL = 8AC5h
LD (HL), A ; (8AC5h) ← 3Ch

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD (HL), n	(HL) ← n	--	--	--	--	3	00 110 110
							← n →

Loads 8-bit immediate data n into memory specified by register pair HL.

Example: When HL = 8AC5h,
LD (HL), 0 ; (8AC5h) ← 0

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD A, (BC)	A ← (BC)	--	--	--	--	2	00 001 010

Loads the contents specified by the contents of register pair BC into register A.

Example: When (BC) = 2Fh,
LD A, (BC) ; A ← 2Fh

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD A, (DE)	A ← (DE)	--	--	--	--	2	00 011 010

Loads the contents specified by the contents of register pair DE into register A.

Example: When (DE) = 5Fh,
LD A, (DE) ; A ← 5Fh

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD A, (C)	A ← (FF00H+C)	--	--	--	--	2	11	110	010

Loads into register A the contents of the internal RAM, port register, or mode register at the address in the range FF00h-FFFFh specified by register C.

Example: When C = 95h,
LD A, (C) ; A ← contents of (FF95h)

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (C), A	(FF00H+C) ← A	--	--	--	--	2	11	100	010

Loads the contents of register A in the internal RAM, port register, or mode register at the address in the range FF00h-FFFFh specified by register C.

Example: When C = 9Fh,
LD (C), A ; (FF9Fh) ← A

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD A, (n)	A ← (n)	--	--	--	--	3	11	110	000
							← n →		

Loads into register A the contents of the internal RAM, port register, or mode register at the address in the range FF00h-FFFFh specified by the 8-bit immediate operand n.

Note, however, that a 16-bit address should be specified for the mnemonic portion of n, because only the lower-order 8 bits are automatically reflected in the machine language.

Example: To load data at FF34h into register A, type the following.
LD A, (FF34)

Typing only LD A, (34) would cause the address to be incorrectly interpreted as 0034, resulting in the instruction LD A, (0034) .

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (n), A	(n) ← A	--	--	--	--	3	11	100	000
							← n →		

Loads the contents of register A to the internal RAM, port register, or mode register at the address in the range FF00h-FFFFh specified by the 8-bit immediate operand n.

Note, however, that a 16-bit address should be specified for the mnemonic portion of n, because only the lower-order 8 bits are automatically reflected in the machine language.

Example: To load the contents of register A in 0xFF34, type the following.
LD (FF34), A

Typing only LD (34), A would cause the address to be incorrectly interpreted as 0034, resulting in the instruction LD (0034), A .

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD A, (nn)	A ← (nn)	--	--	--	--	4	11 111 010
							← n →
							← n →

Loads into register A the contents of the internal RAM or register specified by 16-bit immediate operand nn.

Example: LD A, (FF44h) ; A ← (LY)
LD A, (8000h) ; A ← (8000h)

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD (nn), A	(nn) ← A	--	--	--	--	4	11 101 010
							← n →
							← n →

Loads the contents of register A to the internal RAM or register specified by 16-bit immediate operand nn.

Example: LD (FF44h), A ; (LY) ← A
LD (8000h), A ; (8000h) ← A

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD A, (HLI)	A ← (HL) HL ← HL+1	--	--	--	--	2	00 101 010

Loads in register A the contents of memory specified by the contents of register pair HL and simultaneously increments the contents of HL.

Example: When HL = 1FFh and (1FFh) = 56h,
LD A, (HLI) ; A ← 56h, HL ← 200h

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD A, (HLD)	A ← (HL) HL ← HL-1	--	--	--	--	2	00 111 010

Loads in register A the contents of memory specified by the contents of register pair HL and simultaneously decrements the contents of HL.

Example: When HL = 8A5Ch and (8A5Ch) = 3Ch,
LD A, (HLD) ; A ← 3Ch, HL ← 8A5Bh

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (BC), A	(BC) ← A	--	--	--	--	2	00	000	010

Stores the contents of register A in the memory specified by register pair BC.

Example: When BC = 205Fh and A = 3Fh,
LD (BC) , A ; (205Fh) ← 3Fh

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (DE), A	(DE) ← A	--	--	--	--	2	00	010	010

Stores the contents of register A in the memory specified by register pair DE.

Example: When DE = 205Ch and A = 00h,
LD (DE) , A ; (205Ch) ← 00h

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (HLI), A	(HL) ← A HL ← HL+1	--	--	--	--	2	00	100	010

Stores the contents of register A in the memory specified by register pair HL and simultaneously increments the contents of HL.

Example: When HL = FFFFh and A = 56h,
LD (HLI), A ; (0xFFFF) ← 56h, HL = 0000h

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (HLD), A	(HL) ← A HL ← HL-1	--	--	--	--	2	00	110	010

Stores the contents of register A in the memory specified by register pair HL and simultaneously decrements the contents of HL.

Example: HL = 4000h and A = 5h,
LD (HLD), A ; (4000h) ← 5h, HL = 3FFFh

4.2.2 16-Bit Transfer Instructions

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD dd, nn	dd ← nn	--	--	--	--	3	00 dd0 001
							← n →
	L-ADRS						
	H-ADRS						← n →

Loads 2 bytes of immediate data to register pair dd.

dd codes are as follows:

Register Pair	dd
BC	00
DE	01
HL	10
SP	11

Example: LD HL, 3A5Bh ; H ← 3Ah, L ← 5Bh

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
LD SP, HL	SP ← HL	--	--	--	--	2	11 111 001

Loads the contents of register pair HL in stack pointer SP.

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
PUSH qq	(SP - 1) ← qqH (SP - 2) ← qqL SP ← SP-2	--	--	--	--	4	11 qq0 101

Pushes the contents of register pair qq onto the memory stack. First 1 is subtracted from SP and the contents of the higher portion of qq are placed on the stack. The contents of the lower portion of qq are then placed on the stack. The contents of SP are automatically decremented by 2.

qq codes are as follows:

Register Pair	qq
BC	00
DE	01
HL	10
AF	11

Example: When SP = FFFEh,
PUSH BC ; (FFFDh) ← B, (FFFCh) ← C, SP ← FFFCh

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
POP qq	qqL ← (SP) qqH ← (SP+1) SP ← SP+2	--	--	--	--	3	11	qq0	001

Pops contents from the memory stack and into register pair qq.
First the contents of memory specified by the contents of SP are loaded in the lower portion of qq. Next, the contents of SP are incremented by 1 and the contents of the memory they specify are loaded in the upper portion of qq. The contents of SP are automatically incremented by 2.

Example: When SP = FFFCh, (FFFCh) = 5Fh, and (FFFDh) = 3Ch,
POP BC ; B ← 3Ch, C ← 5Fh, SP ← FFFEh

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LDHL SP, e	HL ← SP+e	*	*	0	0	3	11	111	000
	* Varies with instruction results						← e →		

e = -128 to +127

The 8-bit operand e is added to SP and the result is stored in HL.

Flag Z: Reset
H: Set if there is a carry from bit 11; otherwise reset.
N: Reset
CY: Set if there is a carry from bit 15; otherwise reset.

Example: When SP = 0xFFF8h,
LDHL SP, 2 ; HL ← 0xFFFAh, CY ← 0, H ← 0, N ← 0, Z ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
LD (nn), SP	(nn) ← SP_L (nn+1) ← SP_H	--	--	--	--	5	00	001	000
		L-ADRS					← n →		
		H-ADRS					← n →		

Stores the lower byte of SP at address nn specified by the 16-bit immediate operand nn and the upper byte of SP at address nn + 1.

Example: When SP = FFF8h,
LD (C100h) , SP ; C100h ← F8h
C101h← FFh

4.2.3 8-Bit Arithmetic and Logical Operation Instructions

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADD A, r	A ← A + r	*	*	0	*	1	10	000	r

Adds the contents of register r to those of register A and stores the results in register A.

Flag
Z: Set if the result is 0; otherwise reset.
H: Set if there is a carry from bit 3; otherwise reset.
N: Reset
CY: Set if there is a carry from bit 7; otherwise reset.

Example: When A = 0x3Ah and B = 0xC6h,
ADD A, B ; A ← 0, Z ← 1, H ← 1, N ← 0, CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADD A, n	A ← A + n	*	*	0	*	2	11	000	110
							◄— n —►		

Adds 8-bit immediate operand n to the contents of register A and stores the results in register A..

Example: When A = 3Ch,
ADD A, 0FFh ; A ← 3Bh, Z ← 0, H ← 1, N ← 0, CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADD A, (HL)	A ← A + (HL)	*	*	0	*	2	10	000	110

Adds the contents of memory specified by the contents of register pair HL to the contents of register A and stores the results in register A..

Example: When A = 3Ch and (HL) = 12h,
ADD A, (HL) ; A ← 4Eh, Z ← 0, H ← 0, N ← 0, CY ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
ADC A, s	A← A+s+CY	*	*	0	*	--	--	--	--

Adds the contents of operand s and CY to the contents of register A and stores the results in register A.. r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
ADC A, r	1	10	001	r
ADC A, n	2	11	001	110
		◄— n —►		
ADC A, (HL)	2	10	001	110

Examples: When A = E1h, E = 0Fh, (HL) = 1Eh, and CY = 1,
ADC A, E ; A ← F1h, Z ← 0, H ← 1, CY ← 0
ADC A, 3Bh ; A ← 1Dh, Z ← 0, H ← 0, CY ← 1
ADC A, (HL) ; A ← 00h, Z ← 1, H ← 1, CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SUB s	A ← A-s	*	*	1	*	--	--	--	--

Subtracts the contents of operand s from the contents of register A and stores the results in register A. r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
SUB r	1	10	010	r
SUB n	2	11	010	110
		◄— n —►		
SUB (HL)	2	10	010	110

Flag Z: Set if result is 0; otherwise reset.
H: Set if there is a borrow from bit 4; otherwise reset.
N: Set
CY: Set if there is a borrow; otherwise reset.

Examples: When A = 3Eh, E = 3Eh, and (HL) = 40h,
SUB E ; A ← 00h, Z ← 1, H ← 0, N ← 1 CY ← 0
SUB 0Fh ; A ← 2Fh, Z ← 0, H ← 1, N ← 1 CY← 0
SUB (HL) ; A ← FEh, Z ← 0, H ← 0, N ← 1 CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SBC A, s	A ← A-s-CY	*	*	1	*	--	--	--	--

Subtracts the contents of operand s and CY from the contents of register A and stores the results in register A.
r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
SBC A, r	1	10	011	r
SBC A, n	2	11	011	110
		← n →		
SBC A. (HL)	2	10	011	110

Examples: When A = 3Bh, (HL) = 4Fh, H = 2Ah, and CY = 1,
SBC A, H ; A ← 10h, Z ← 0, H ← 0, N ← 1 CY ← 0
SBC A, 3Ah ; A ← 00h, Z ← 1, H ← 0, N ← 1 CY ← 0
SBC A, (HL) ; A ← EBh, Z ← 0, H ← 1, N ← 1 CY ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
AND s	A ← A ∧ s	0	1	0	*	--	--	--	--

Takes the logical-AND for each bit of the contents of operand s and register A, and stores the results in register A.
r, n, and (HL) are used for operand s.

	CYCL	7 6	5 4 3	2 1 0
AND r	1	10	100	r
AND n	2	11	100	110
		← n →		
AND (HL)	2	10	100	110

Examples: When A = 5Ah, L = 3Fh and (HL) = 0h,
AND L ; A ← 1Ah, Z ← 0, H ← 1, N ← 0 CY ← 0
AND 38h ; A ← 18h, Z ← 0, H ← 1, N ← 0 CY ← 0
AND (HL) ; A ← 00h, Z ← 1, H ← 1, N ← 0 CY ← 0

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
OR s	AVs	0	0	0	*	--	-- -- --

Takes the logical-OR for each bit of the contents of operand s and register A and stores the results in register A. r, n, and (HL) are used for operand s.

	CYCL	7 6 5 4 3 2 1 0
OR r	1	10 110 r
OR n	2	11 110 110
		← n →
OR (HL)	2	10 110 110

Examples: When A = 5Ah, (HL) = 0Fh,
OR A ; A ← 5Ah, Z ← 0
OR 3 ; A ← 5Bh, Z ← 0
OR (HL) ; A ← 5Fh, Z ← 0

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
XOR s	A ⊕ s	0	0	0	*	--	-- -- --

Takes the logical exclusive-OR for each bit of the contents of operand s and register A. and stores the results in register A. r, n, and (HL) are used for operand s.

	CYCL	7 6 5 4 3 2 1 0
XOR r	1	10 101 r
XOR n	2	11 101 110
		← n →
XOR (HL)	2	10 101 110

Examples: When A = FFh and (HL) = 8Ah,
XOR A ; A ← 00h, Z ← 1
XOR 0x0Fh ; A ← F0h, Z ← 0
XOR (HL) ; A ← 75h, Z ← 0

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
CP s	A — s	*	*	1	*	--	-- -- --

Compares the contents of operand s and register A and sets the flag if they are equal.
r, n, and (HL) are used for operand s.

	CYCL	7 6 5 4 3 2 1 0
CP r	1	10 111 r
CP n	2	11 111 110
		← n →
CP (HL)	2	10 111 110

Examples: When A = 3Ch, B = 2Fh, and (HL) = 40h,
CP B ; Z ← 0, H ← 1, N ← 1, CY ← 0
CP 3Ch ; Z ← 1, H ← 0, N ← 1, CY ← 0
CP (HL) ; Z ← 0, H ← 0, N ← 1, CY ← 1

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
INC r	r ← r + 1	--	*	0	*	1	00 r 100

Increments the contents of register r by 1.

Example: When A = FFh,
INC A ; A ← 0, Z ← 1, H ← 1, N ← 0

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
INC (HL)	(HL) ← (HL) + 1	--	*	0	*	3	00 110 100

Increments by 1 the contents of memory specified by register pair HL.

Example: When (HL) = 0x50h,
INC (HL) ; (HL) ← 0x51h, Z ← 0, H ← 0, N ← 0

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
DEC r	r ← r - 1	--	*	1	*	1	00 r 101

Subtract 1 from the contents of register r by 1.

Example: When L = 01h,
DEC L ; L ← 0, Z ← 1, H ← 0, N ← 1

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
DEC (HL)	(HL) ← (HL) - 1	--	*	1	*	3	00	110	101

Decrements by 1 the contents of memory specified by register pair HL.

Example: When (HL) = 00h,
DEC (HL) ; (HL) ← FFh, Z ← 0, H ← 1, N ← 1

4.2.4 16-Bit Arithmetic Operation Instructions

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
ADD HL, ss	HL ← HL + ss	*	*	0	--	2	00 ss1 001

Adds the contents of register pair ss to the contents of register pair HL and stores the results in HL.
ss codes are as follows:

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

Flag Z: No change
H: Set if there is a carry from bit 11; otherwise reset.
N: Rest
CY: Set if there is a carry from bit 15; otherwise reset.

Example: When HL = 8A23h, BC = 0605h,
ADD HL, BC ; HL ← 9028h, H ← 1, N ← 0, CY ← 0
ADD HL, HL ; HL ← 1446h, H ← 1, N ← 0, CY ← 1

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
ADD SP, e	SP ← SP + e	*	*	0	0	4	11 101 000
							◄— e —►

Adds the contents of the 8-bit immediate operand e and SP and stores the results in SP.

Example: SP = FFF8h
ADD SP, 2 ; SP ← FFFAh, CY ← 0, H ← 0, N ← 0, Z ← 0

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
INC ss	ss ← ss + 1	--	--	--	--	2	00 ss0 011

Increments the contents of register pair ss by 1.

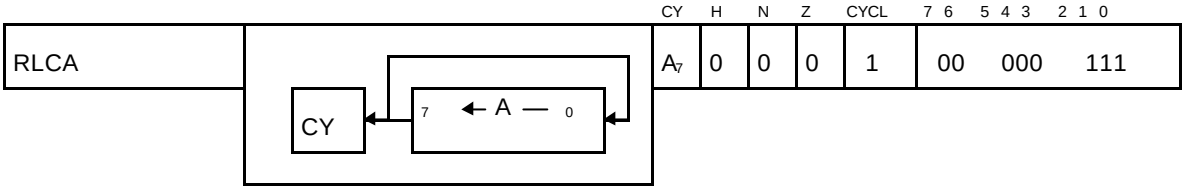
Example: When DE = 235Fh,
INC DE ; DE ← 2360h

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
DEC ss	ss ← ss - 1	--	--	--	--	2	00 ss1 011

Decrements the contents of register pair ss by 1.

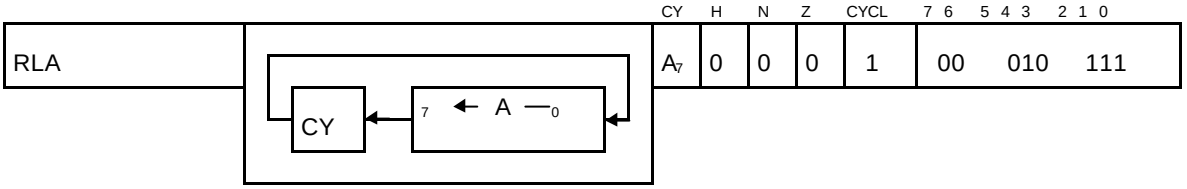
Example: When DE = 235Fh,
DEC DE ; DE ← 235Eh

4.2.5 Rotate/Shift Instructions



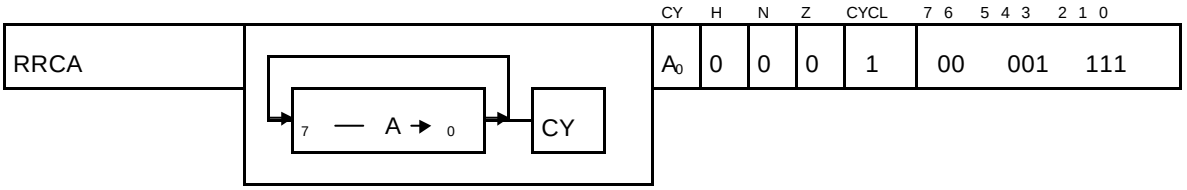
Rotates the contents of register A to the left.
That is, the contents of bit 0 are copied to bit 1 and the previous contents of bit 1 (the contents before the copy operation) are copied to bit 2. The same operation is repeated in sequence for the rest of the register. The contents of bit 7 are placed in both CY and bit 0 of register A..

Example: When A = 85h and CY = 0,
RLCA ; A ← 0Ah, CY ← 1, Z ← 0, H ← 0, N ← 0



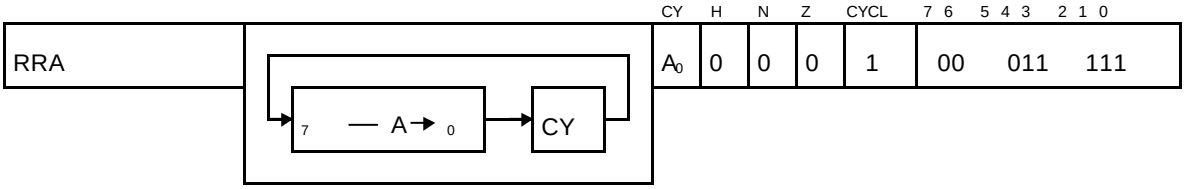
Rotates the contents of register A to the left.

Example: When A = 95h and CY = 1,
RLA ; A ← 2Bh, CY ← 1, Z ← 0, H ← 0, N ← 0



Rotates the contents of register A to the right.

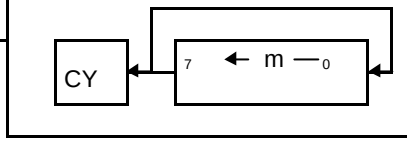
Example: When A = 3Bh and CY = 0,
RRCA ; A ← 9Dh, CY ← 1, Z ← 0, H ← 0, N ← 0



Rotates the contents of register A to the right.

Example: When A = 81h and CY = 0,
RRA ; A ← 40h, CY ← 1, Z ← 0, H ← 0, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RLC	m	m_7	0	0	*	--	--	--	--

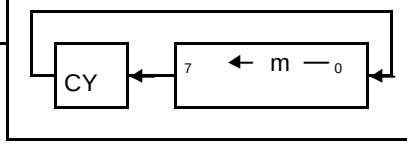


Rotates the contents of operand m to the left.
r and (HL) are used for operand m.

		CYCL	7 6	5 4 3	2 1 0
RLC	r	2	11	001	011
			00	000	r
RLC	(HL)	4	11	001	011
			00	000	110

Examples: When B = 85h, (HL) = 0, and CY = 0,
RLC B ; B ← 0Bh, CY ← 1, Z ← 0, H ← 0, N ← 0
RLC (HL) ; (HL) ← 00h, CY ← 0, Z ← 1, H ← 0, N ← 0

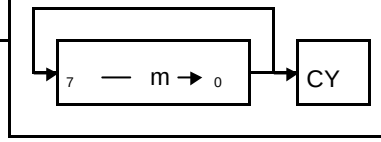
		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RL	m	m_7	0	0	*	--	--	--	--



Rotates the contents of operand m to the left.
r and (HL) are used for operand m.

		CYCL	7 6	5 4 3	2 1 0
RL	r	2	11	001	011
			00	010	r
RL	(HL)	4	11	001	011
			00	010	110

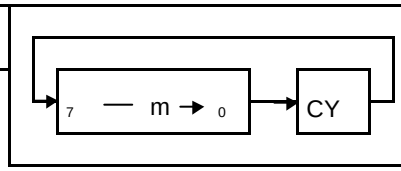
Examples: When L = 80h, (HL) = 11h, and CY = 0,
RL L ; L ← 00h, CY ← 1, Z ← 1, H ← 0, N ← 0
RL (HL) ; (HL) ← 22h, CY ← 0, Z ← 0, H ← 0, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RRC m		m_0	0	0	*	--	--	--	--

Rotates the contents of operand m to the right.
r and (HL) are used for operand m.

	CYCL	7 6	5 4 3	2 1 0
RRC r	2	11	001	011
		00	001	r
RRC (HL)	4	11	001	011
		00	001	110

Examples: When C = 1h, (HL) = 0h, CY = 0,
RRC C ; C ← 80h, CY ← 1, Z ← 0, H ← 0, N ← 0
RRC (HL) ; (HL) ← 00h, CY ← 0, Z ← 1, H ← 0, N ← 0

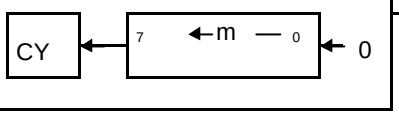
		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RR m		m_0	0	0	*	--	--	--	--

Rotates the contents of operand m to the right.
r and (HL) are used for operand m.

	CYCL	7 6	5 4 3	2 1 0
RR r	2	11	001	011
		00	011	r
RR (HL)	4	11	001	011
		00	011	110

Examples: When A = 1h, (HL) = 8Ah, CY = 0,
RR A ; A ← 00h, CY ← 1, Z ← 1, H ← 0, N ← 0
RR (HL) ; (HL) ← 45h, CY ← 0, Z ← 0, H ← 0, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SLA	m	m_7	0	0	*	--	--	--	--



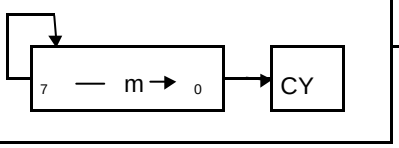
Shifts the contents of operand m to the left. That is, the contents of bit 0 are copied to bit 1 and the previous contents of bit 1 (the contents before the copy operation) are copied to bit 2. The same operation is repeated in sequence for the rest of the operand. The content of bit 7 is copied to CY, and bit 0 is reset.

r and (HL) are used for operand m.

		CYCL	7 6	5 4 3	2 1 0
SLA	r	2	11	001	011
			00	100	r
SLA	(HL)	4	11	001	011
			00	100	110

Examples: When D = 80h, (HL) = FFh, and CY = 0,
SLA D ; D ← 00h, CY ← 1, Z ← 1, H ← 0, N ← 0
SLA (HL) ; (HL) ← FEh, CY ← 1, Z ← 0, H ← 0, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SRA	m	m_0	0	0	*	--	--	--	--

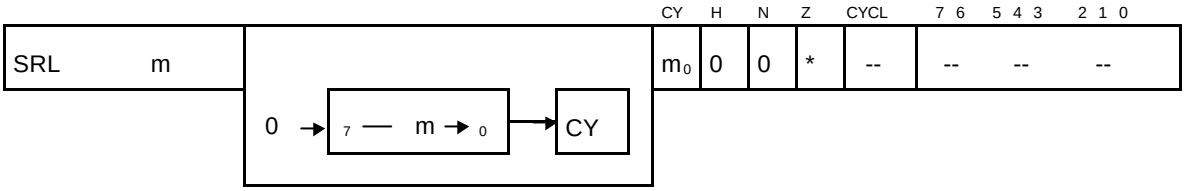


Shifts the contents of operand m to the right. That is, the contents of bit 7 are copied to bit 6 and the previous contents of bit 6 (the contents before the copy operation) are copied to bit 5. The same operation is repeated in sequence for the rest of the operand . The contents of bit 0 are copied to CY, and the content of bit 7 is unchanged.

r and (HL) are used for operand m.

		CYCL	7 6	5 4 3	2 1 0
SRA	r	2	11	001	011
			00	101	r
SRA	(HL)	4	11	001	011
			00	101	110

Example: When A = 8Ah, (HL) = 01h, and CY = 0,
SRA D ; A ← C5h, CY ← 0, Z ← 0, H ← 0, N ← 0
SRA (HL) ; (HL) ← 00h, CY ← 1, Z ← 1, H ← 0, N ← 0

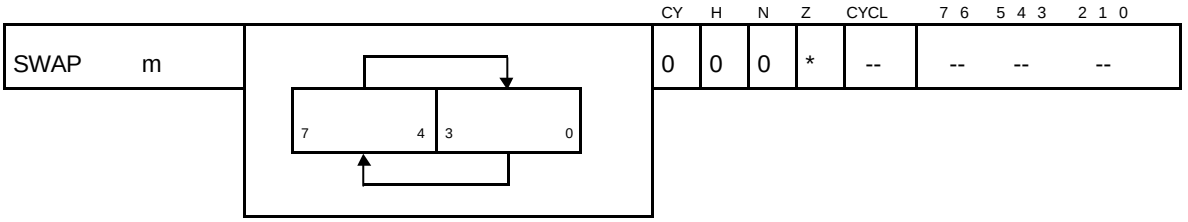


Shifts the contents of operand m to the right. That is, the contents of bit 7 are copied to bit 6 and the previous contents of bit 6 (the contents before the copy operation) are copied to bit 5. The same operation is repeated in sequence for the rest of the operand . The contents of bit 0 are copied to CY, and bit 7 is reset.

r and (HL) are used for operand m.

	CYCL	7 6	5 4 3	2 1 0
SRL r	2	11	001	011
		00	111	r
SRL (HL)	4	11	001	011
		00	111	110

Examples: When A = 01h, (HL) = FFh, CY + 0,
SRL A ; A ← 00h, CY ← 1, Z ← 1, H ← 0, N ← 0
SRL (HL) ; (HL) ← 7Fh, CY ← 1, Z ← 0, H ← 0, N ← 0



Shifts the contents of the lower-order 4 bits (0-3) of operand m unmodified to the higher-order 4 bits (4-7) of that operand and shifts the contents of the higher-order 4 bits to the lower-order 4 bits.
r and (HL) are used for operand m.

	CYCL	7 6	5 4 3	2 1 0
SWAP r	2	11	001	011
		00	110	r
SWAP (HL)	4	11	001	011
		00	110	110

Examples: When A = 00h and (HL) = F0h,
SWAP A ; A ← 00h, Z ← 1, H ← 0, N ← 0, CY ← 0
SWAP (HL) ; (HL) ← 0Fh, Z ← 0, H ← 0, N ← 0, CY ← 0

4.2.6 Bit Operations

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
BIT b, r	Z ← $\overline{r_b}$	--	1	0	$\overline{r_b}$	2	11	001	011
							01	b	r

Copies the complement of the contents of the specified bit in register r to the Z flag of the program status word (PSW).

The codes for b and r are as follows.

Bit	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

Examples: When A = 80h and L = EFh
BIT 7, A ; Z ← 0, H ← 1, N ← 0
BIT 4, L ; Z ← 1, H ← 1, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
BIT b, (HL)	Z ← $\overline{(HL)_b}$	--	1	0	$\overline{(HL)_b}$	3	11	001	011
							01	b	110

Copies the complement of the contents of the specified bit in memory specified by the contents of register pair HL to the Z flag of the program status word (PSW).

Examples: When (HL) = FEh,
BIT 0, (HL) ; Z ← 1, H ← 1, N ← 0
BIT 1, (HL) ; Z ← 0, H ← 1, N ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SET b, r	$r_b \leftarrow 1$	--	--	--	--	2	11	001	011
							11	b	r

Sets to 1 the specified bit in specified register r.

Example: When A = 80h and L = 3Bh,
SET 3, A ; A ← 0x84h
SET 7, L ; L ← 0xBBh

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SET b, (HL)	$(HL)_b \leftarrow 1$	--	--	--	--	4	11	001	011
							11	b	110

Sets to 1 the specified bit in the memory contents specified by registers H and L.

Example: When 00h is the memory contents specified by H and L,
SET 3, (HL) ; (HL) ← 04H

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RES b, r	$r_b \leftarrow 0$	--	--	--	--	2	11	001	011
							10	b	r

Resets to 0 the specified bit in the specified register r.

Example: When A = 80h and L = 3Bh,
RES 7, A ; A ← 00h
RES 1, L ; L ← 39h

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RES b, (HL)	$(HL)_b \leftarrow 0$	--	--	--	--	4	11	001	011
							10	b	110

Resets to 0 the specified bit in the memory contents specified by registers H and L.

Example: When 0xFFh is the memory contents specified by H and L,
RES 3, (HL) ; (HL) ← F7h

4.2.7 Jump Instructions

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
JP nn	PC ← nn	--	--	--	--	4	11 000 011
							L - ADRS ← n →
							H - ADRS ← n →

Loads the operand nn to the program counter (PC).
nn specifies the address of the subsequently executed instruction.
The lower-order byte is placed in byte 2 of the object code and the higher-order byte is placed in byte 3.

Example: JP 8000h ; Jump to 8000h.

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
JP cc, nn	If cc true, PC ← nn	--	--	--	--	4/3	11 Occ 010
							L - ADRS ← n →
							H - ADRS ← n →

*Cycle no. is 3 when cc does not match

Loads operand nn in the PC if condition cc and the flag status match.
The subsequent instruction starts at address nn.
If condition cc and the flag status do not match, the contents of the PC are incremented, and the instruction following the current JP instruction is executed.

The relation between conditions and cc codes are as follows.

Cc	Condition	Flag
00	NZ	Z = 0
01	Z	Z = 1
10	NC	CY = 0
11	C	CY = 1

Example: When Z = 1 and C = 0,

```
JP  NZ,  8000h  ;      Moves to next instruction after 3 cycles.
JP    Z,  8000h  ;     Jumps to address 8000h.
JP    C, 8000h  ;      Moves to next instruction after 3 cycles.
JP  NC,  8000h ;       Jumps to address 8000h.
```

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
JR e	PC ← PC + e	--	--	--	--	3	00 011 000
							← e - 2 →

e = -127 to +129

Jumps -127 to +129 steps from the current address.

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
JR cc, e	If cc true, PC ← PC + e	--	--	--	--	3/2	00 1cc 000
							← e - 2 →

e = -127 to +129

If condition cc and the flag status match, jumps -127 to +129 steps from the current address. If cc and the flag status do not match, the instruction following the current JP instruction is executed.

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
JP (HL)	PC ← HL	--	--	--	--	1	11 101 001

Loads the contents of register pair HL in program counter PC.
The next instruction is fetched from the location specified by the new value of PC.

Example: When HL = 8000h,
JP (HL) ; Jumps to 8000h.

4.2.8 Call and Return Instructions

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
CALL nn	(SP - 1) ← PC_H (SP - 2) ← PC_L PC ← nn SP ← SP-2	--	--	--	--	6	11 001 101
						L - ADRS	← n →
						H - ADRS	← n →

In memory, pushes the PC value corresponding to the instruction at the address following that of the CALL instruction to the 2 bytes following the byte specified by the current SP. Operand nn is then loaded in the PC.

The subroutine is placed after the location specified by the new PC value.
When the subroutine finishes, control is returned to the source program using a return instruction and by popping the starting address of next instruction, which was just pushed, and moving it to the PC.

With the push, the current value of the SP is decremented by 1, and the higher-order byte of the PC is loaded in the memory address specified by the new SP value. The value of the SP is then again decremented by 1, and the lower-order byte of the PC is loaded in the memory address specified by that value of the SP.

The lower-order byte of the address is placed in byte 2 of the object code, and the higher-order byte is placed in byte 3.

Examples: When PC = 8000h and SP = FFFEh,

```
Address
8000h     CALL  1234H      ;      Jumps to address 1234h, and
8003h                              (FFFDH) ← 80H
                                   (FFFCH) ← 03H
                                   SP ← FFFCH
```

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
CALL cc, nn	If cc true, (SP - 1) ← PC_H (SP - 2) ← PC_L PC ← nn SP ← SP – 2	--	--	--	--	6/3	11 Occ 100
						L-ADRS	← n →
						H-ADRS	← n →

If condition cc matches the flag, the PC value corresponding to the instruction following the CALL instruction in memory is pushed to the 2 bytes following the memory byte specified by the SP. Operand nn is then loaded in the PC.

Examples: When Z = 1,

```
Address
7FFCh     CALL  NZ,  1234h     ;      Moves to next instruction after 3 cycles.
8000h      CALL    Z,  1234h     ;      Pushes 8003h to the stack,
8003h                                    and jumps to 1234h.
```

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP + 1)$ $SP \leftarrow SP + 2$	--	--	--	--	4	11	001	001

Pops from the memory stack the PC value pushed when the subroutine was called, returning control to the source program.

In this case, the contents of the address specified by the SP are loaded in the lower-order byte of the PC, and the content of the SP is incremented by 1. The contents of the address specified by the new SP value are then loaded in the higher-order byte of the PC, and the SP is again incremented by 1. (The value of SP is 2 larger than before instruction execution.)

The next instruction is fetched from the address specified by the content of PC.

```
Examples:   Address
            8000H   CALL  9000H
            8003H
            9000H
                    RET             ;  Returns to address 0x8000h
```

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RETI	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP + 1)$ $SP \leftarrow SP + 2$	--	--	--	--	4	11	011	001

Used when an interrupt-service routine finishes.
The execution of this return is as follows.

The address for the return from the interrupt is loaded in program counter PC.
The master interrupt enable flag is returned to its pre-interrupt status.

```
Examples:   0040h
                    RETI            ;  Pops the stack and returns to address 8001h.
            8000H     INC  L        :An external interrupt occurs here.
            8001H
```

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
RET	If cc true, $PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$ $SP \leftarrow SP + 2$	--	--	--	--	5/2	11	Occ	000

If condition cc and the flag match, control is returned to the source program by popping from the memory stack the PC value pushed to the stack when the subroutine was called.

```
Example:    Address
            8000h   CALL    9000h
            8003h

9000h       CP      0
            RET     Z               ; Returns to address 8000h if Z = 1.
                                      Moves to next instruction after 2 cycles if Z = 0.
```

		CY	H	N	Z	CYCL	7 6 5 4 3 2 1 0
RST t	(SP - 1) ← PC_H (SP - 2) ← PC_L SP ← SP - 2 PC_H ← 0 PC_L ← P	--	--	--	--	4	11 t 111

Pushes the current value of the PC to the memory stack and loads to the PC the page 0 memory addresses provided by operand t.
Then next instruction is fetched from the address specified by the new content of PC.

With the push, the content of the SP is decremented by 1, and the higher-order byte of the PC is loaded in the memory address specified by the new SP value. The value of the SP is then again decremented by 1, and the lower-order byte of the PC is loaded in the memory address specified by that value of the SP.

The RST instruction can be used to jump to 1 of 8 addresses.

Because all of the addresses are held in page 0 memory, 0x00h is loaded in the higher-order byte of the PC, and the value of P is loaded in the lower-order byte.

The relation between the t codes and P are as follows.

Operand	t	(PC_H)	P (PC_L)
0	000	00h	00h
1	001	00h	08h
2	010	00h	10h
3	011	00h	18h
4	100	00h	20h
5	101	00h	28h
6	110	00h	30h
7	111	00h	38h

Example:
```
Address
8000h    RST    1    ;  Pushes 8001h to the stack ,
8001h                    and jumps to 0008h.
```

4.2.9 General-Purpose Arithmetic Operations and CPU Control Instructions

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
DAA	Decimal adjust acc	*	0	--	*	1	00	100	111

When performing addition and subtraction, binary coded decimal representation is used to set the contents of register A to a binary coded decimal number (BCD).
The following table shows the processing that accompanies execution of the DAA instruction immediately following execution of addition (ADD and ADC)and substraction (SUB and SBC) instructions.

Instruction Before Execution	CY Contents before Execution	Bits 4-7 Register A	H Contents before Execution	Bits 0-3 Register A	Number Added to Register A	CY Contents after Execution
	0	0h- 9h	0	0h- 9h	00h	0
	0	0h- 8h	0	Ah-Fh	06h	0
ADD	0	0h- 9h	1	0h- 3h	06h	0
ADC	0	Ah-Fh	0	0h- 9h	60h	1
	0	9h-Fh	0	Ah-Fh	66h	1
	0	Ah-Fh	1	0h- 3h	66h	1
	1	0h- 2h	0	0h- 9h	60h	1
	1	0h- 2h	0	Ah-Fh	66h	1
(N = 0)	1	0h- 3h	1	0h- 3h	66h	1
SUB	0	0h-9h	0	0h- 9h	00h	0
SBC	0	0h-8h	1	6h- Fh	FAh	0
	1	7h-Fh	0	0h- 9h	A0h	1
(N = 1)	1	6h-Fh	1	6h- Fh	9Ah	1

Examples: When A = 45h and B = 38h,
ADD A, B ; A ← 7Dh, N ← 0
DAA ; A ←7Dh + 06h (83h), CY ← 0
SUB A, B ; A ← 83h – 38h (4Bh), N ← 1
DAA ; A ← 4Bh + FAh (45h) , CY ← 0

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
CPL	$A \leftarrow \overline{A}$	--	1	1	--	1	00	101	111

Takes the one's complement of the contents of register A.

Example: When A = 35h,
CPL ; A ← CAh

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
NOP	No operation	--	--	--	--	1	00	000	000

Only advances the program counter by 1; performs no other operations that have an effect.

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
CCF	CY ← $(\overline{CY})$	$\overline{(CY)}$	0	0	--	1	00	111	111

Flips the carry flag CY.

Example: When CY = 1,
```
CCF         ; CY ← 0
```

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
SCF	CY ← 1	1	0	0	--	1	00	110	111

Sets the carry flag CY.

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
DI	IME ← 0	--	--	--	--	1	11	110	011

Resets the interrupt master enable flag and prohibits maskable interrupts.

Note ***Even if a DI instruction is executed in an interrupt routine, the IME flag is set if a return is performed with a RETI instruction.***

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
EI	IME ← 1	--	--	--	--	1	11	111	011

Sets the interrupt master enable flag and enables maskable interrupts.
This instruction can be used in an interrupt routine to enable higher-order interrupts.

Note ***The IME flag is reset immediately after an interrupt occurs. The IME flag reset remains in effect if control is returned from the interrupt routine by a RET instruction. However, if an EI instruction is executed in the interrupt routine, control is returned with IME = 1.***

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
HALT	Halt	--	--	--	--	1	01	110	110

After a HALT instruction is executed, the system clock is stopped and HALT mode is entered. Although the system clock is stopped in this status, the oscillator circuit and LCD controller continue to operate.

In addition, the status of the internal RAM register ports remains unchanged.

HALT mode is canceled by an interrupt or reset signal.

The program counter is halted at the step after the HALT instruction. If both the interrupt request flag and the corresponding interrupt enable flag are set, HALT mode is exited, even if the interrupt master enable flag is not set.

Once HALT mode is canceled, the program starts from the address indicated by the program counter.

If the master enable flag is set, the contents of the program counter are pushed to the stack and control jumps to the starting address of the interrupt.

If the RESET terminal goes LOW in HALT mode, the mode becomes that of a normal reset.

		CY	H	N	Z	CYCL	7 6	5 4 3	2 1 0
STOP	Stop	--	--	--	--	1	00	010	000
							00	000	000

Execution of a STOP instruction stops both the system clock and oscillator circuit. STOP mode is entered, and the LCD controller also stops.

However, the status of the internal RAM registers ports remains unchanged.

STOP mode can be canceled by a reset signal.

If the RESET terminal goes LOW in STOP mode, it becomes that of a normal reset status.

The following conditions should be met before a STOP instruction is executed and STOP mode is entered.

- All interrupt-enable (IE) flags are reset.
- Input to P10 — P13 is LOW for all.

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 5: MISCELLANEOUS GENERAL INFORMATION

5.1 Monitor ROM 131

5.2 Recognition Data for CGB (CGB only) in ROM-registered Data 132

5.3 Power-Saving Routines for the Main Program 133

5.4 Software Created Exclusively for CGB 135

5.5 Software Created to Operate on CGB 136

5.6 Software Created to Operate on CGB: Example 137

5.6.1 Program Specifications 137
5.6.2 CGB Recognition Method 138
5.6.3 Flowcharts 139

THIS PAGE WAS INTENTIONALLY LEFT BLANK

CHAPTER 5: MISCELLANEOUS GENERAL INFORMATION

5.1 Monitor ROM

The DMG and CGB CPU includes internal monitor ROM.

When power on the hardware is turned on, the monitor ROM checks for errors in the 'Nintendo' logo character data within the game software.

If the data is correct, the Nintendo logo is displayed and the program is then started. If there is an error in the data, the screen flashes repeatedly.

For information on registering the Nintendo logo character data, refer to Appendix 3 of this manual, *Submission Requirements*.

The conditions required for starting the user program are as follows.

Starting Address:	150h (default value)	The starting address can be freely set by writing a jump destination address at 102h and 103h.
LCDC value:	91h	
Stack value:	FFFEh	

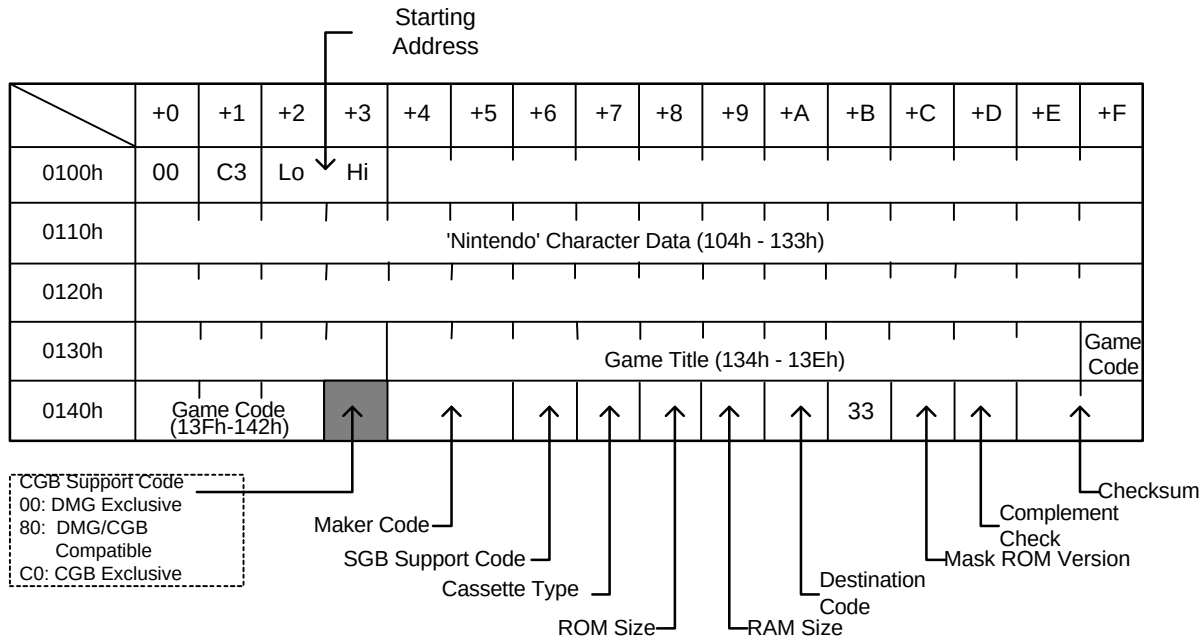
5.2 Recognition Data for CGB (CGB only) in ROM-Registered Data

As with software created for DMG, software for CGB (including software only for CGB) must place data concerning items such as the name of the game and Game Pak specifications in the 80 bytes of the program area between 100h and 14Fh. In the system, a code indicating whether the software is for CGB should be set at address 143h.

> ***Note:*** ***For an overall description of the ROM area shown below, please refer to Appendix 3,* Submission Requirements.**

Setting a value of 80h or C0h at this address causes the system to recognize the software as being for CGB.

If 00h or any value less than 7Fh (existing DMG software) is set at this address, the software is recognized as non-CGB software and CGB functions (registers) are not available.



CGB/CGB Only: When operating on CGB, up to 56 colors can be displayed on a single screen.
Non-CGB: When operating on CGB, up to 10 colors can be displayed on a single screen.

> ***Note:*** ***Regardless of the type of game, the following fixed values should be stored at the following addresses.***

- Address 100h=00h
- Address 101h=C3h
- Address 14Bh=33h
- Addresses 104h – 133h='Nintendo' character data

5.3 Power-Saving Routines for the Main Program

To minimize battery power consumption and extend battery life, inclusion of programs such as those shown below is recommended.

During waiting for vertical blanking, halt the CPU system clock to reduce power consumption by the CPU and ROM.

```
;******                                          ******
;******          Main Routine                    ******
;******                                          ******

MAIN
            CALL    CONT            :   Keypad input.
            CALL    GAME            :   Game or other processing.

VBLK_WT
            HALT                    :   Halt the system clock.
                                    :   Return from HALT mode if an interrupt is generated.
                                    :   Wait for a vertical blanking interrupt.
            NOP                     :   Used to avoid bugs in the rare case that the
                                        instruction. after the HALT instruction is not
                                        executed.
            LD      A, (VBLK_F)
            AND     A               :   Generate a V-blank interrupt?
            JR      Z, VBLK_WT      :   Jump if a non-V-blank interrupt.
            XOR     A
            LD      (VBLK_F), A
            JR      MAIN

;******                                          ******
;******    Vertical Blanking Routine             ******
;******                                          ******

VBLK
            PUSH    AF
            PUSH    BC
            PUSH    DE
            PUSH    HL

            CALL    DMA

            LD      A, 1            :   Set the V-blank completion flag.
            LD      (VBLK_F), A

            POP     HL
            POP     DE
            POP     BC
            POP     AF
            RETI
```

HALT instructions should not be executed while CGB horizontal blanking DMA is executed. (See Appendix 1, *Programming Cautions*.)

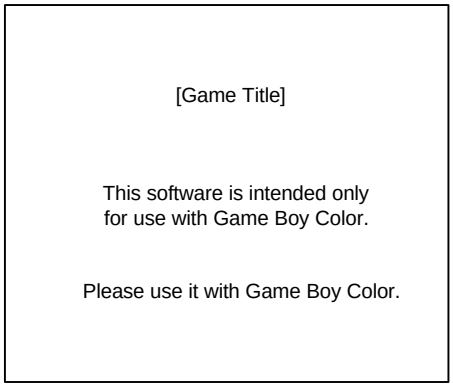
5.4 Software Created Exclusively for CGB

Because the shape of the Game Pak for CGB-only software is the same as that for DMG, CGB-only Game Paks also can be inserted in DMG. Therefore, a program that displays a message such as that shown below when a CGB-only Game Pak is mistakenly inserted in DMG should always be included in the software. The upper part of the message screen should display the official title of the game.

If the title is similar to that of other software (e.g., series software), a subtitle should also be displayed to distinguish the programs from one another.

For information on software methods of distinguishing game units, see Section 6 of this chapter, *Software Created for CGB: Example*.

Sample Message Display



5.5 Software Created to Operate on CGB

As is shown below, CGB and DMG differ slightly in their specifications and operation. When creating software to operate on CGB, please give appropriate consideration to these differences.

CGB	DMG
When objects with different x-coordinates overlap, the object with the lowest OBJ NO. is given display priority.	When objects with different x-coordinates overlap, the object with the smallest x-coordinate is given display priority.
In CGB mode, BG display CANNOT be turned off using bit 0 of the LCDC register (address FF40h).	BG display CAN be turned on and off using bit 0 of the LCDC register (address FF40h).
———	When the value of register WX (address FF4Bh) is 166, the window is partially displayed.
———	When an instruction that register pair increment is used, if the value of the register pair is an address that specifies OAM (FE00h-FE9Fh), OAM may be destroyed.

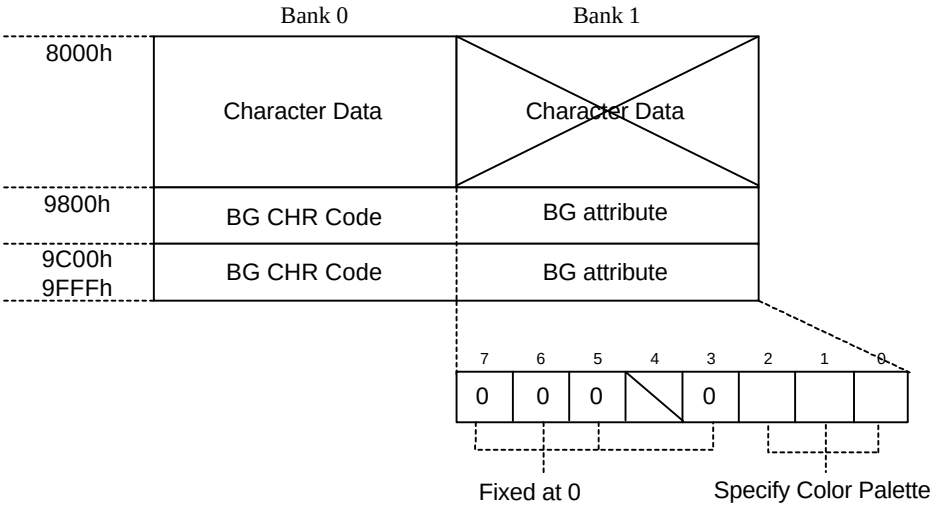
5.6 Software Created to Operate on CGB: Example

When creating software for CGB, a CGB support code is set in the ROM data area, and processing branches according to the hardware used internally by the program. For more information, see the flowchart in Part 1 of Section 6.3 of this chapter. Limiting the functions used, as shown below, allows the same processing to be used for different units without branching. For more information, see the flowchart in Part 2 of Section 6.3 of this chapter.

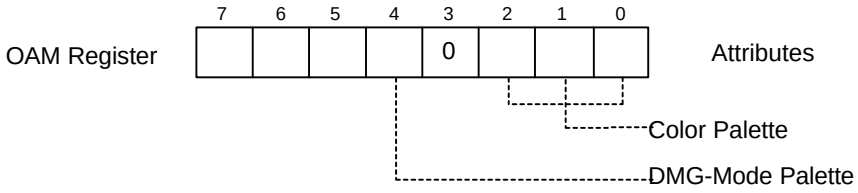
The following example describes how to create a program that operates on both CGB and DMG and allows display of 56 colors when running on CGB . Such means can be used to maintain compatibility with earlier hardware (DMG) while using CGB functions.

5.6.1 Program Specifications

- Only bank 0 is used as the character data area.
- Only the bits that specify the color palette (bits 0-2 of bank 1) are used for BG attributes.



- Both the color palette and DMG-mode palette are set as attribute flags in the OAM register.



- None of the other expanded CGB functions are used.

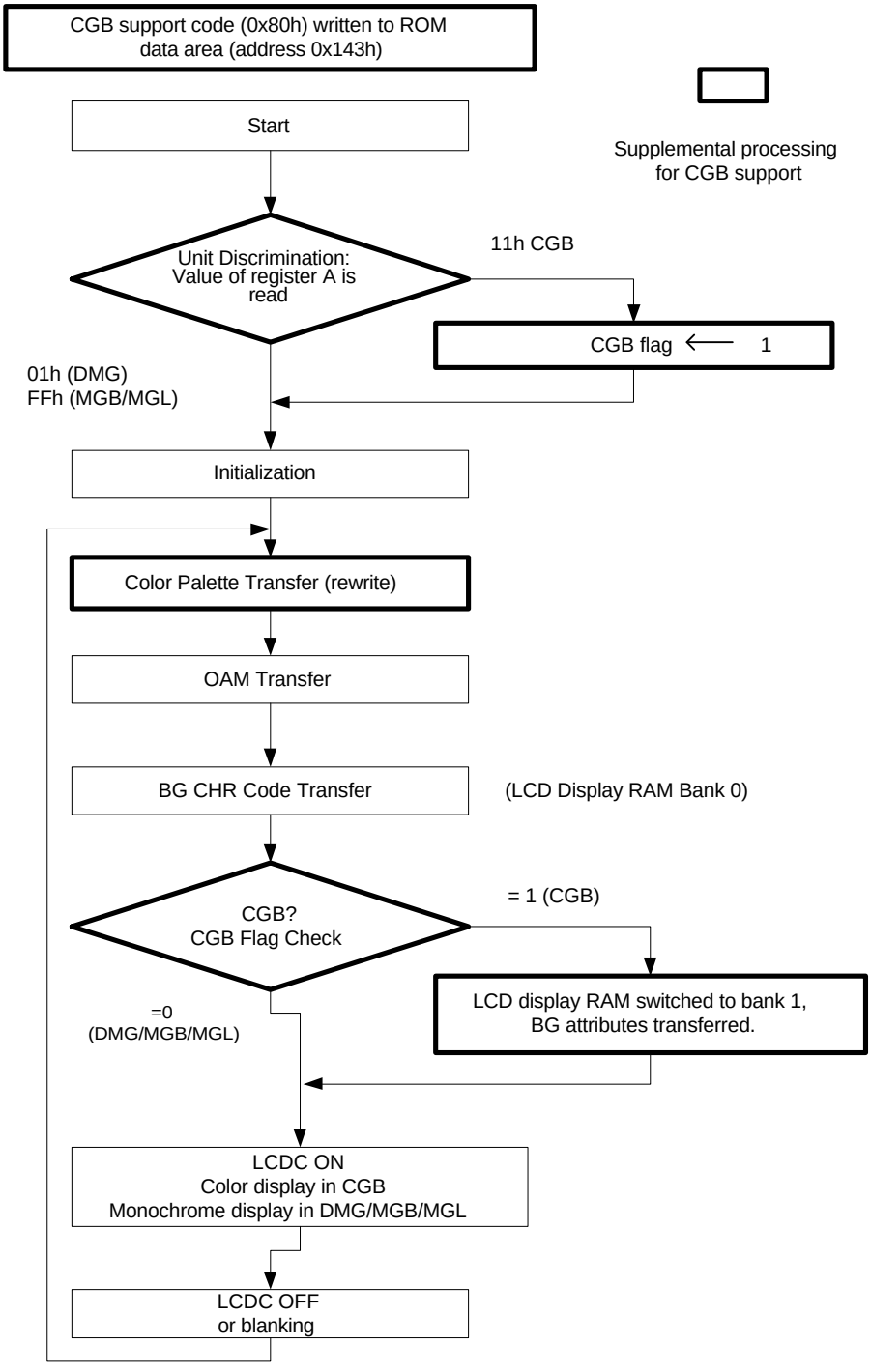
5.6.2 CGB Recognition Method

Immediately after program startup, the initial value of the accumulator (register A) is read to determine whether the hardware on which the program is operating is a DMG (SGB), MGB/MGL (SGB2), or CGB.

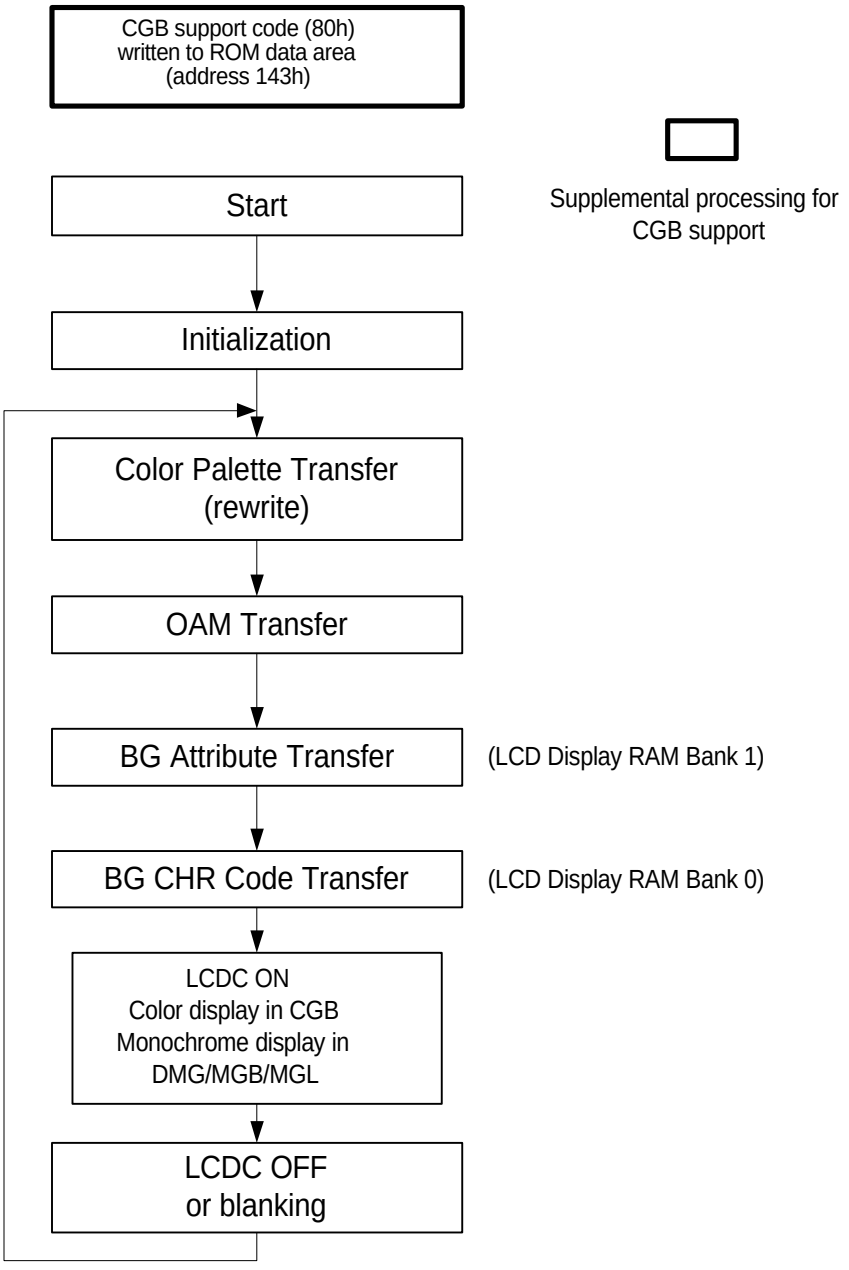
01h $\rightarrow$ DMG (SGB)
FFh $\rightarrow$ MGB/MGL (SGB2)
11h $\rightarrow$ CGB

5.6.3 Flowcharts

1) Branched Processing for CGB and DMG/MGB/MGL


CGB support code (0x80h) written to ROM data area (address 0x143h)
Start
Supplemental processing for CGB support
Unit Discrimination: Value of register A is read
11h CGB
CGB flag ← 1
01h (DMG)
FFh (MGB/MGL)
Initialization
Color Palette Transfer (rewrite)
OAM Transfer
BG CHR Code Transfer
(LCD Display RAM Bank 0)
CGB? CGB Flag Check
= 1 (CGB)
LCD display RAM switched to bank 1, BG attributes transferred.
=0 (DMG/MGB/MGL)
LCDC ON
Color display in CGB
Monochrome display in DMG/MGB/MGL
LCDC OFF or blanking

2) Uniform processing for CGB and DMG/MGB/MGL



Note: ***The BG attributes should always be transferred before the BG character code.***

Even if only the BG attributes are changed, always transfer the character code from that same address.

CHAPTER 6: THE SUPER GAME BOY SYSTEM 143

6.1 Overview 143

6.1.1 What is Super Game Boy?.....143
6.1.2 Block Diagram.....144
6.1.3 Functions 144
6.1.4 System Program 145

6.2 Sending Commands and Data to SUPER NES..... 146

6.2.1 System Commands.....146
6.2.2 Data Transfer Using an Image Signal 149

6.3 System Commands 150

6.3.1 System Command Summary.....150
6.3.2 System Command Details 151
6.3.3 Cautions Regarding Sending Commands.....188
6.3.4 Sound Flag Summary.....188

6.4 Miscellaneous 194

6.4.1 Reading Input from Multiple Controllers 194
6.4.2 Recognizing SGB 195
6.4.3 SGB Register Summary.....197
6.4.4 Flowchart of Initial Settings Routine 198

6.5 Programming Cautions..... 199

6.5.1 ROM Registration Data 199
6.5.2 Initial Data.....199
6.5.3 SOU_TRN initial data.....199

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 6: THE SUPER GAME BOY SYSTEM

6.1 Overview

6.1.1 What is Super Game Boy?

SGB is a device that enables Game Boy software to be enjoyed on a TV screen. Game Boy software can be plugged into the SGB, which operates on the Super Nintendo Entertainment System.

SGB consists of the basic Game Boy circuitry, and components such as an Intercommunication Device (ICD, with built-in SGB RAM), the system program, and a CIC.

Basic SGB operation involves conversion by the ICD of 2-bit, 4 grayscale image signals generated by the SGB CPU to Super NES character data and storage of these data in SGB RAM. The system program subsequently transfers this data by DMA to Super NES W-RAM and then to V-RAM. The above operations are performed repeatedly to display the Game Boy screen on a TV screen.

Unmodified sound output from the SGB CPU is linked to the Super NES sound mixing circuit and is output from the speaker on the TV.

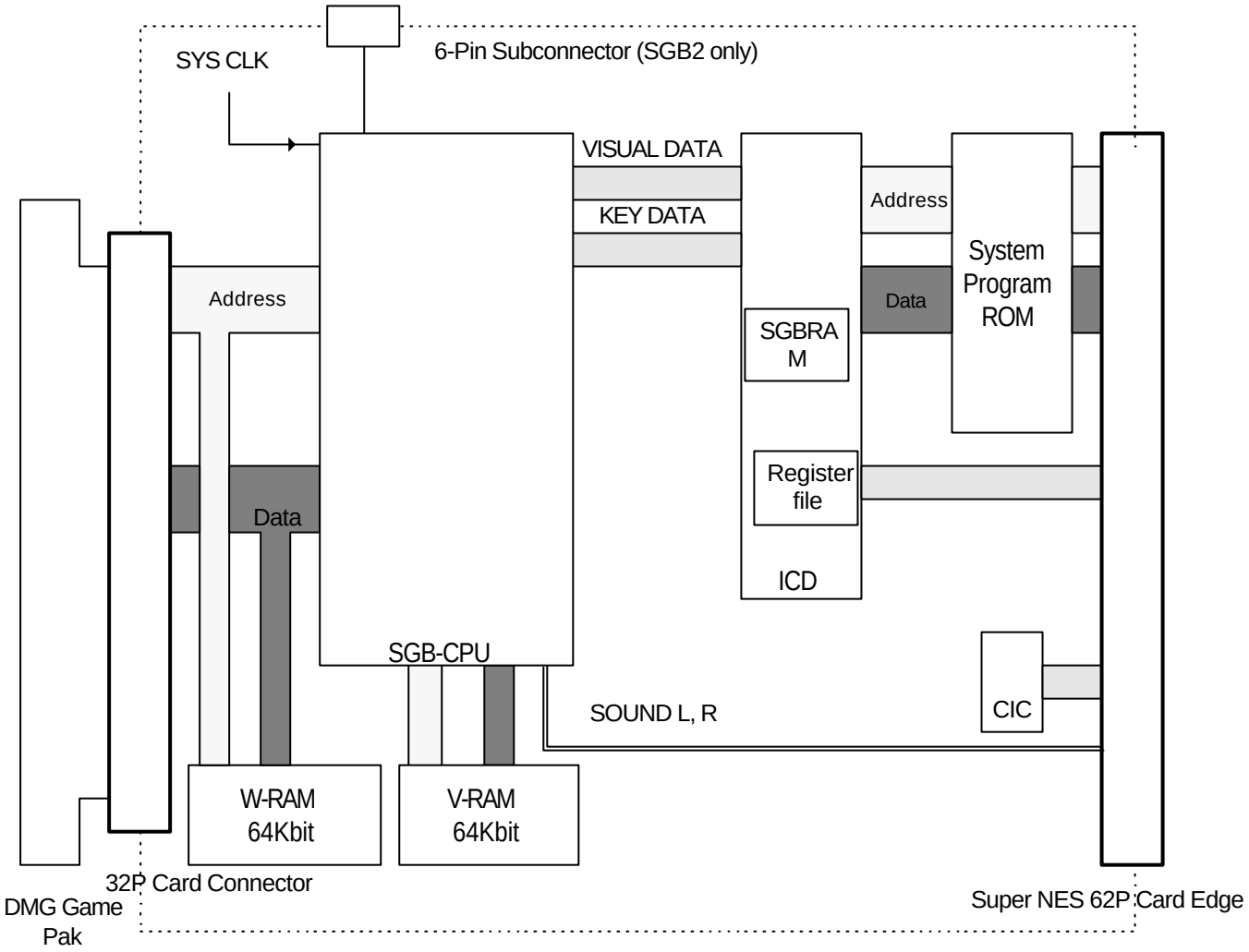
These operations are controlled by the SGB system program and therefore require no special consideration when programming for Game Boy.

Game Boy software not specifically created for SGB provides 4 colors in 4 grayscales. These colors are selected from several color patterns provided in the system program. Programming using the system commands described later allows a game to be represented using 4 palettes of 4 colors each per screen and Super NES functions such as Super NES sound.

Super Game Boy comes in 2 models: the 1994 model, which has no communication connector, and the 1998 model, which is equipped with a communication connector.

This manual uses the term SGB2 when discussing points that concern only the 1998 model. Descriptions that use the term Super Game Boy or SGB refer to both Super Game Boy models. SGB2 allows game representations that use Super NES functions for communication play. (SGB2 has not been released in the U.S. market.)

6.1.2 Block Diagram



6.1.3 Functions

The types of representations indicated below can be implemented using Super NES functions invoked by sending system commands.

For more information, please see Section 3 in this chapter, *System Commands*.

Image Functions

Up to 4 palettes of 4 colors each can be represented on a single screen.

Multiple areas can be specified for each screen, and separate color palette attributes can be specified for each area.

Color palette attributes can be specified separately for each character (8 x 8 bits).

Sound Functions
The rich variety of sound effects included in the system program can be generated by the Super NES audio processing unit (APU).

The sound generator included in the system program can be used by transferring music data.

Controller Functions
Data from multiple Super NES controllers data can be read, providing for multiplayer games that can accommodate between 2 to 4 players.

Miscellaneous
Super NES program data can be transferred.

6.1.4 System Program
The system program can provide the following features.

On the T.V. screen, the system program displays the space outside the game screen (picture frame).

The picture frame has the following features.
- The frame can be selected from among 9 pre-loaded frames.
- A mode in which an image created by the game producer is transferred and displayed as the frame.
- A drawing mode that allows the user to create the frame.

Features of the color palette selection screen are as follows.
- Palettes can be selected from among 32 pre-loaded palettes.
- A mode that allows colors to be set from DMG in DMG games.

A mode is available that allows the user to arrange the colors on a palette.

A screen is provided for changing the key configuration of the controller.

If the commands described in Section 3.2 in this chapter, *System Command Details*, are sent to the register file, Super NES functions, such as those described in Section 1.3, *Functions*, can be used by having the system program read these commands.

6.2 Sending Commands and Data to Super NES

The following 2 methods can be used to send data from a DMG program to Super NES.

Send data to the register file using P14 and P15. The size of the register file is 128 bits; this is referred to as 1 packet.

Send data to SGB RAM using an image signal.

> ***Note:*** ***Data transfers from the register file and SGB RAM to Super NES are performed by the system program.***

6.2.1 System Commands

Using the register file to transmit system commands allows the various Super NES functions described below to be used in games.

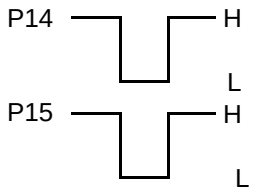
The system program receives the commands and performs the specified processing.

Data Format of System Commands

1) Data Transmission Methods
   Using 2 bits in SGB (P14 and P15 of SGB CPU), data is sent to the register file by serial transmission.

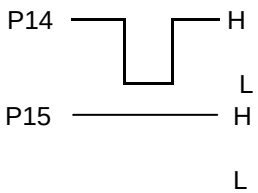
   The system program reads the contents written to the register file.

   1. Start write



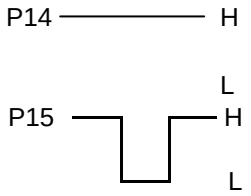
A LOW pulse is output to both P14 and P15.
This is required for transmission of each packet (128 bits).

   2. Write 0



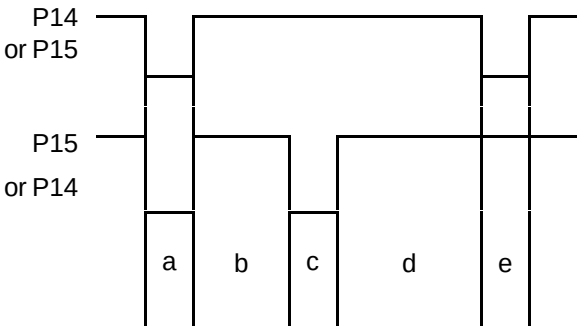
P15 is fixed at HIGH, and a LOW pulse is output to P14.

Write 1



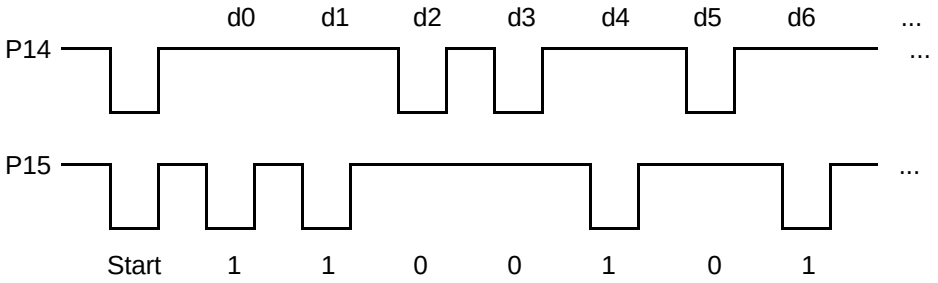
P14 is fixed at HIGH, and
a LOW pulse is output to P15.

2) Pulse Width

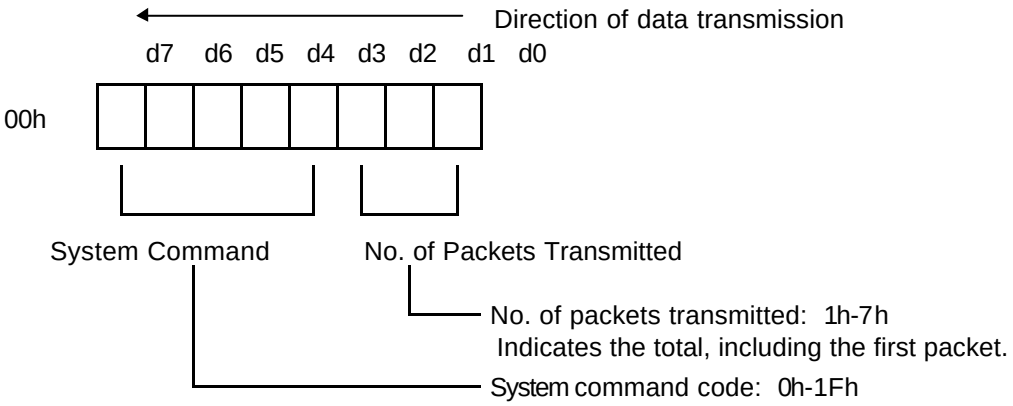


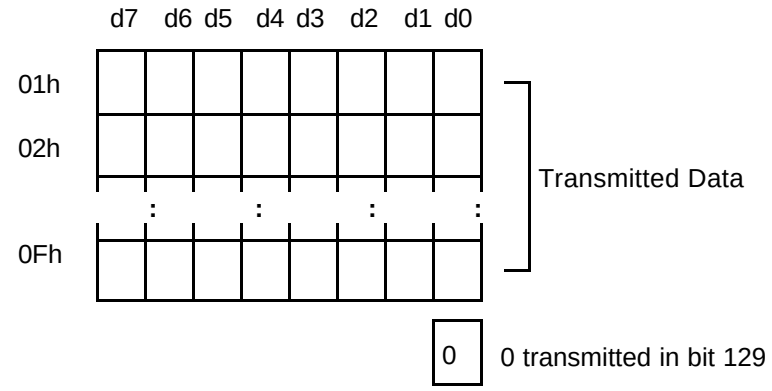
Pulse Width	
a, c, e	5 µ s (min)
b, d	15 µ s (min)

3) Write Example



4) Format of Data Transmitted to Register File


Direction of data transmission
d7 d6 d5 d4 d3 d2 d1 d0
00h
System Command
No. of Packets Transmitted
No. of packets transmitted: 1h-7h
Indicates the total, including the first packet.
System command code: 0h-1Fh

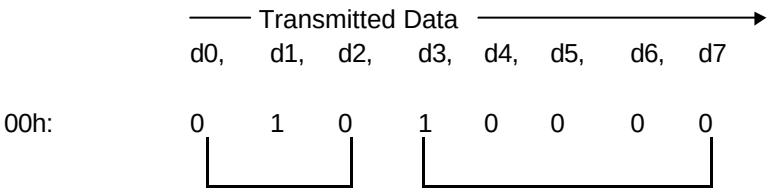


If 2 or more packets are used for one system command, bits 00h-Fh of the second packet onward are used for data.

Transmission Procedure

1. Start of write

2. Data transmission (example)



No. of packets: 2h Command code: 1h

01h:	data
02h:	data
:	:
:	:
Fh:	data

3. Transmission of 0 in bit 129

Bit 129: 0

4. Start of write

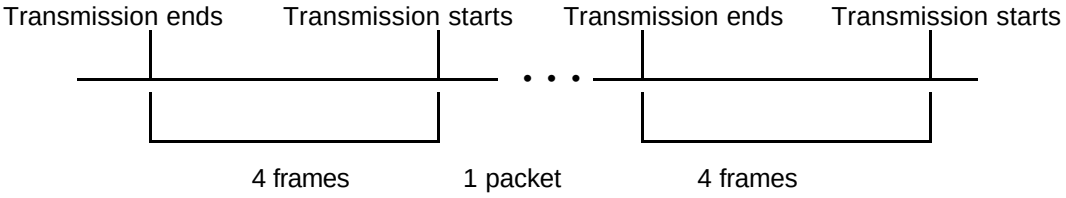
5. Data transmission: second packet

00h:	data
01h:	data
:	:
:	:
Fh:	data

6. Transmission of 0 in bit 129

Bit 129: 0

5) Transmission Interval
The interval between completion of transmission of one packet (128 bits + 1 bit) and transmission of the next packet is set at approximately 60 msec (4 frames).



6) Transmission Bit 129
The data in bit 129 marks the end of one packet, so it should always be transmitted.

6.2.2 Data Transfer Using an Image Signal

Data and programs stored in a Game Pak can be transferred using the image signal transmission path (LD0, LD1).

Character data stored in DMG VRAM and displayed are then stored in SGB RAM. The system program usually transfers these data to Super NES VRAM as character data. However, when a specific command is received, the data is handled as data for command processing.

The displayed image signal is handled directly as data, so be careful to ensure that the OBJ display and window are set to OFF, the correct values are set for the DMB color palette, and the BG to be displayed is correctly transferred.

When data is transferred they are displayed to the screen, so the system command MASK_EN must be used to mask the screen.

For more information, see Section 3.2 in this chapter, *System Command Details*.

Note: ***Commands that transfer data using image signals are indicated by the heading, Data Transfer Using VRAM.***

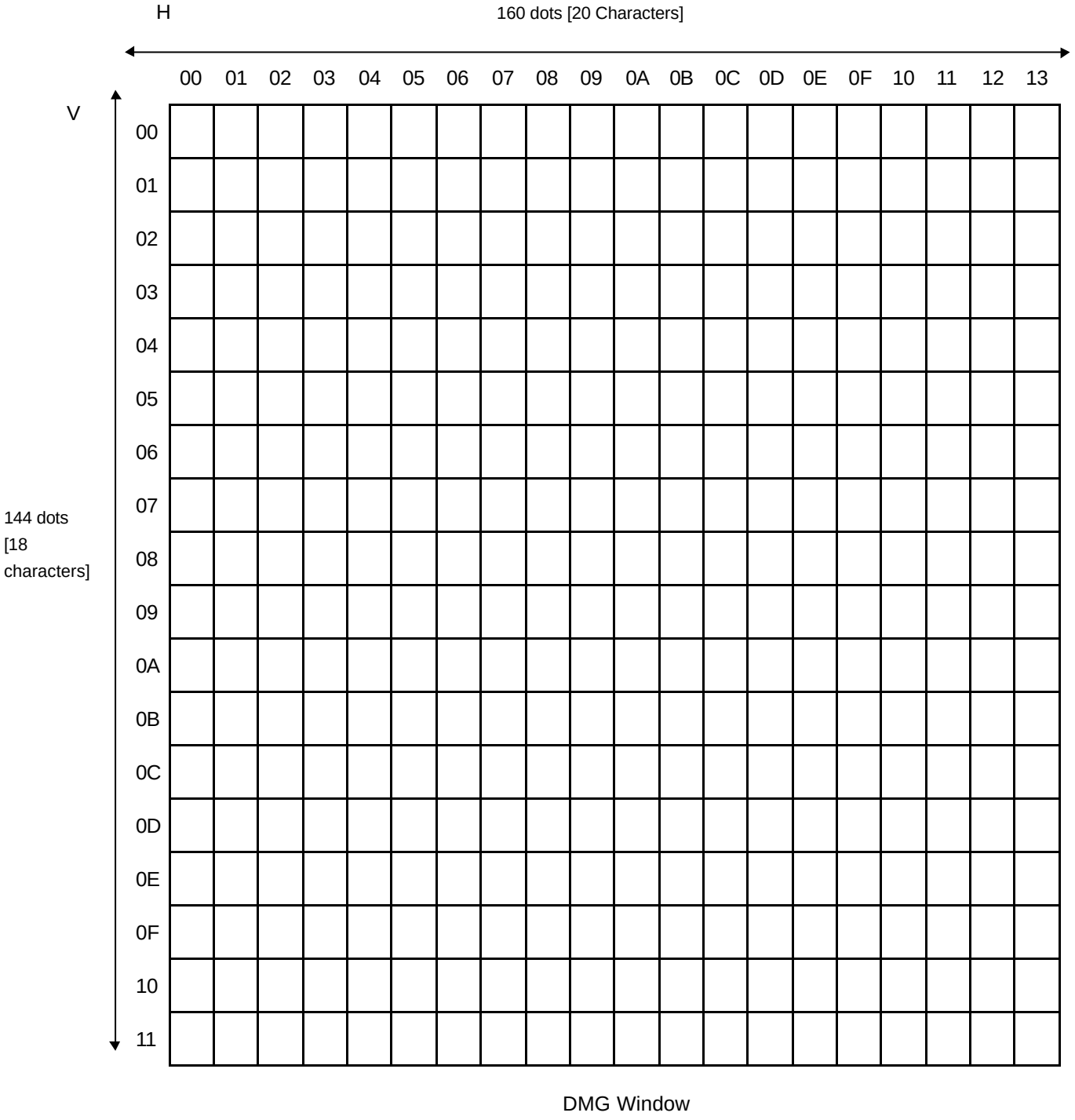
6.3 System Commands

6.3.1 System Command Summary

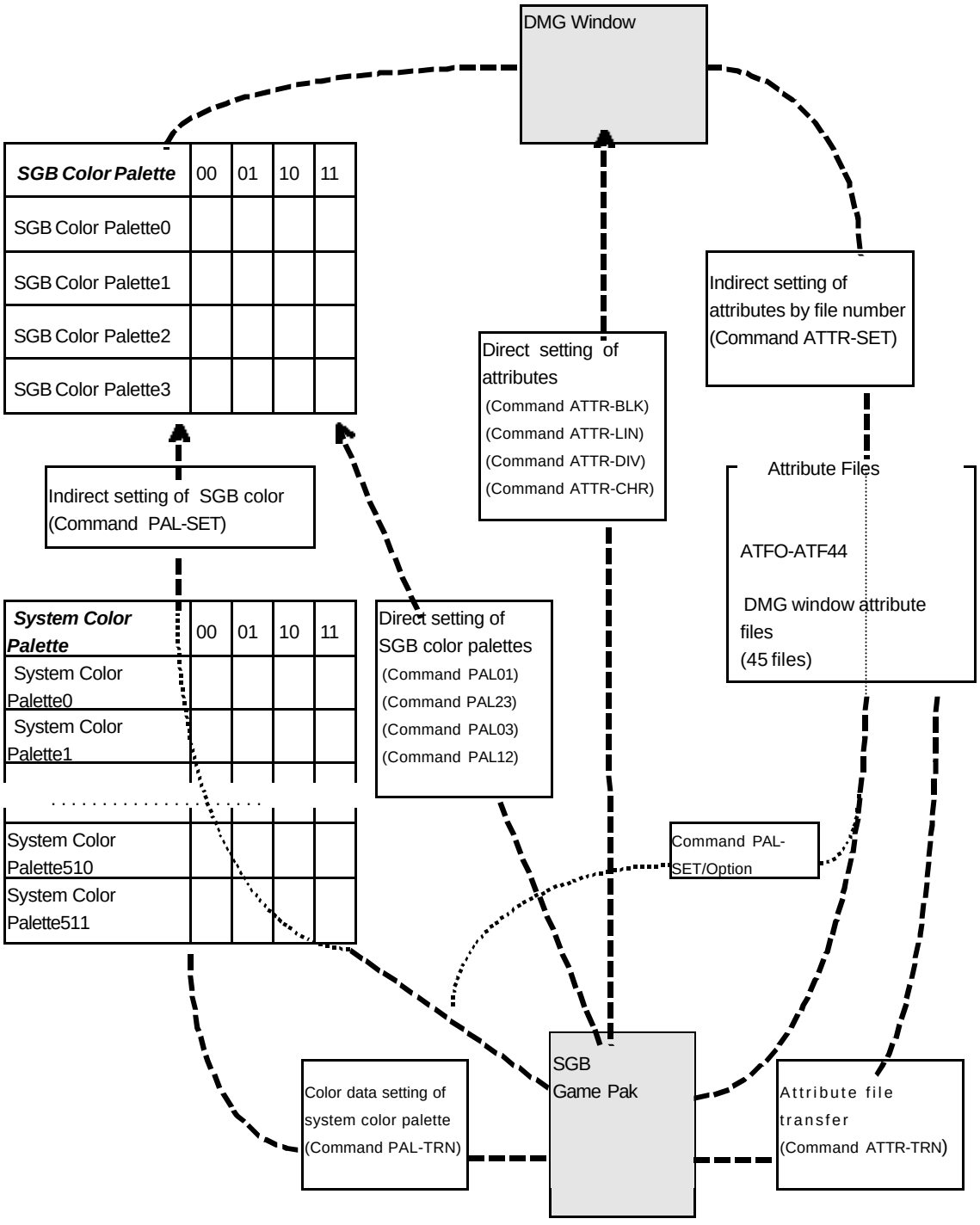
Command	Command Code	Command	Command Code
PAL01	00	DATA_TRN	10
PAL23	01	MLT_REQ	11
PAL03	02	JUMP	12
PAL12	03	CHR_TRN	13
ATTR_BLK	04	PCT_TRN	14
ATTR_LIN	05	ATTR_TRN	15
ATTR_DIV	06	ATTR_SET	16
ATTR_CHR	07	MASK_EN	17
SOUND	08	PAL_PRI	19
SOU_TRN	09	Use prohibited	0D
PAL_SET	0A	Use prohibited	18
PAL_TRN	0B		
ATRC_EN	0C		
ICON_EN	0E		
DATA_SND	0F		

6.3.2 System Command Details

Please refer to the following map in the discussion of coordinate settings and color palette area specifications in the description of the system command functions.



Setting the Color Palettes and Attributes



Note: ***Bit 00 of SGB color palettes 0 – 3 have the same color. The color setting in effect for this bit is the most recent setting.***

DMG Color Palettes and SGB Color Palettes

With DMG screen data representations, colors in SGB are converted from the grayscale data registered in the DMG color palettes, rather than being converted from the bit data for the character.

DMG palette

00	01	10	11	(m,n)
00	01	10	11	

* 00-11 in the palette are grayscale data

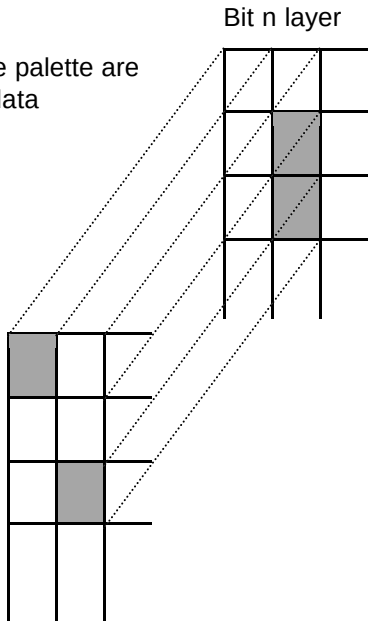
light 00> 01> 10> 11 dark

Table 1

SGB palette

00	01	10	11
Red	Green	Gray	Black

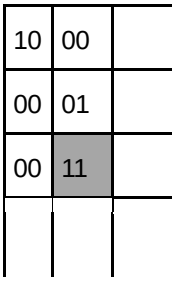
Table 2



Example: When the grayscale data shown in Table 1 are specified for the DMG palette, the character represented on the DMG LCD is as shown in the DMG character image figure below and to the right.

Accordingly, when the color data shown in Table 2 are specified for the SGB palette, the character image represented on Super NES is as shown in the SGB character image figure below and to the right. .....

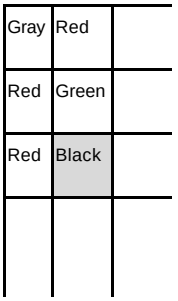
DMG Character Image



00-11: grayscale data

However, if bit 11 of the DMG palette is set to grayscale 00, the ▒ portion of the DMG character image is displayed with a 00 grayscale, and the ▒ portion of the SGB character image is displayed as red rather than black.

SGB Character Image

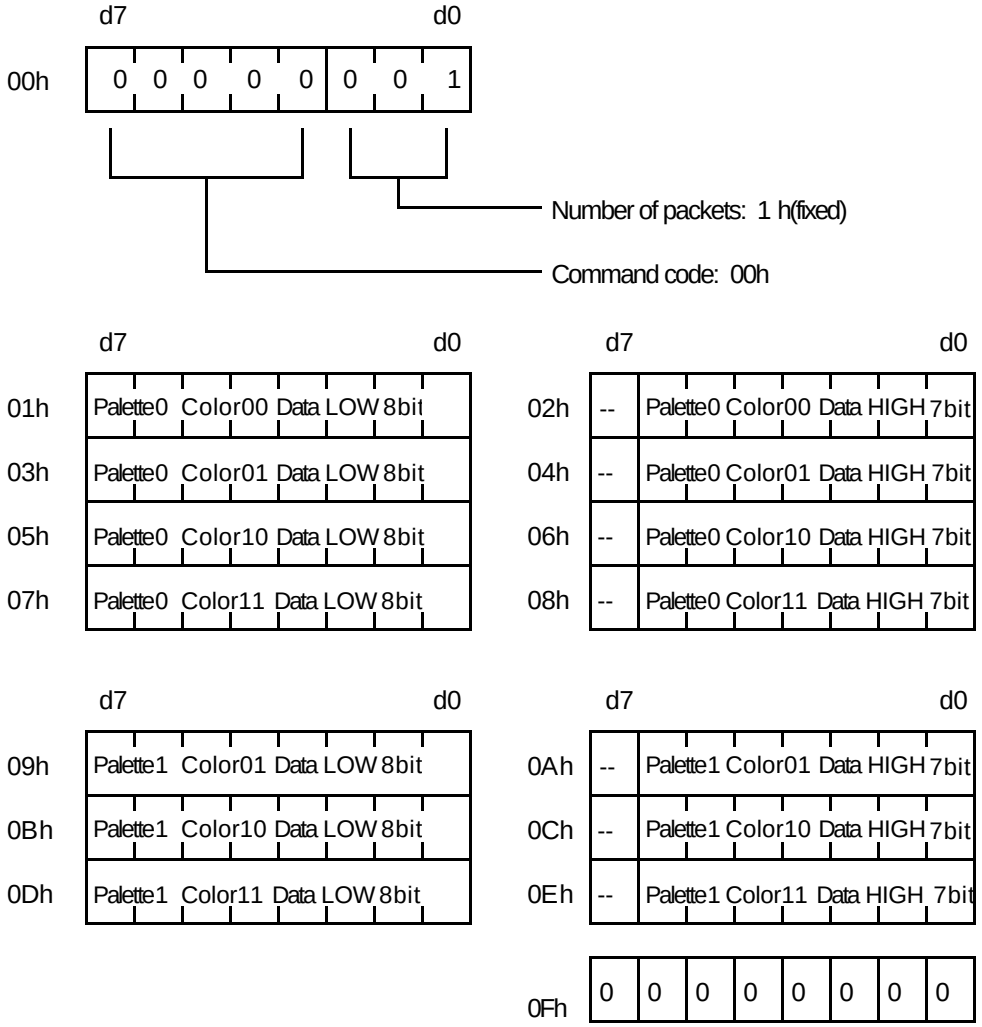


Thus, in this case, when character data display using all of the colors on the SGB palette is desired, a separate grayscale palette (DMG palette) for SGB must be provided, DMG and SGB must be distinguished, and the program must be made to branch accordingly.
(See Section 6.4.2, *Recognizing SGB*.)

When representing DMG grayscale on SGB, the image can be faithfully represented if 00 of the SGB palette is set to a light color and 11 to a dark color.

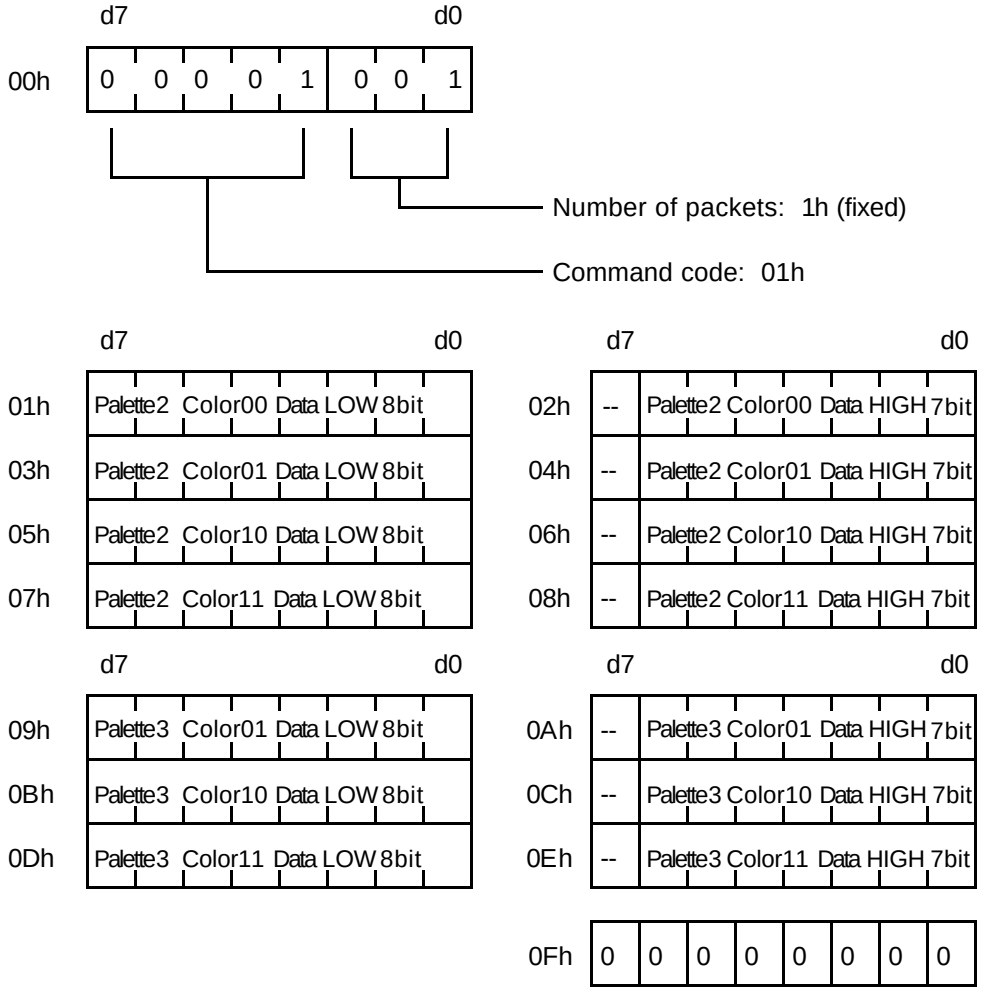
Command: PAL01 (Code: 00h)

Function: Sets the color data of SGB color palettes 0 and 1.


d7
d0
00h
0 0 0 0 0 0 0 1
Number of packets: 1 h(fixed)
Command code: 00h
01h Palette0 Color00 Data LOW 8bit
03h Palette0 Color01 Data LOW 8bit
05h Palette0 Color10 Data LOW 8bit
07h Palette0 Color11 Data LOW 8bit
02h -- Palette0 Color00 Data HIGH 7bit
04h -- Palette0 Color01 Data HIGH 7bit
06h -- Palette0 Color10 Data HIGH 7bit
08h -- Palette0 Color11 Data HIGH 7bit
09h Palette1 Color01 Data LOW 8bit
0Bh Palette1 Color10 Data LOW 8bit
0Dh Palette1 Color11 Data LOW 8bit
0Ah -- Palette1 Color01 Data HIGH 7bit
0Ch -- Palette1 Color10 Data HIGH 7bit
0Eh -- Palette1 Color11 Data HIGH 7bit
0Fh 0 0 0 0 0 0 0 0

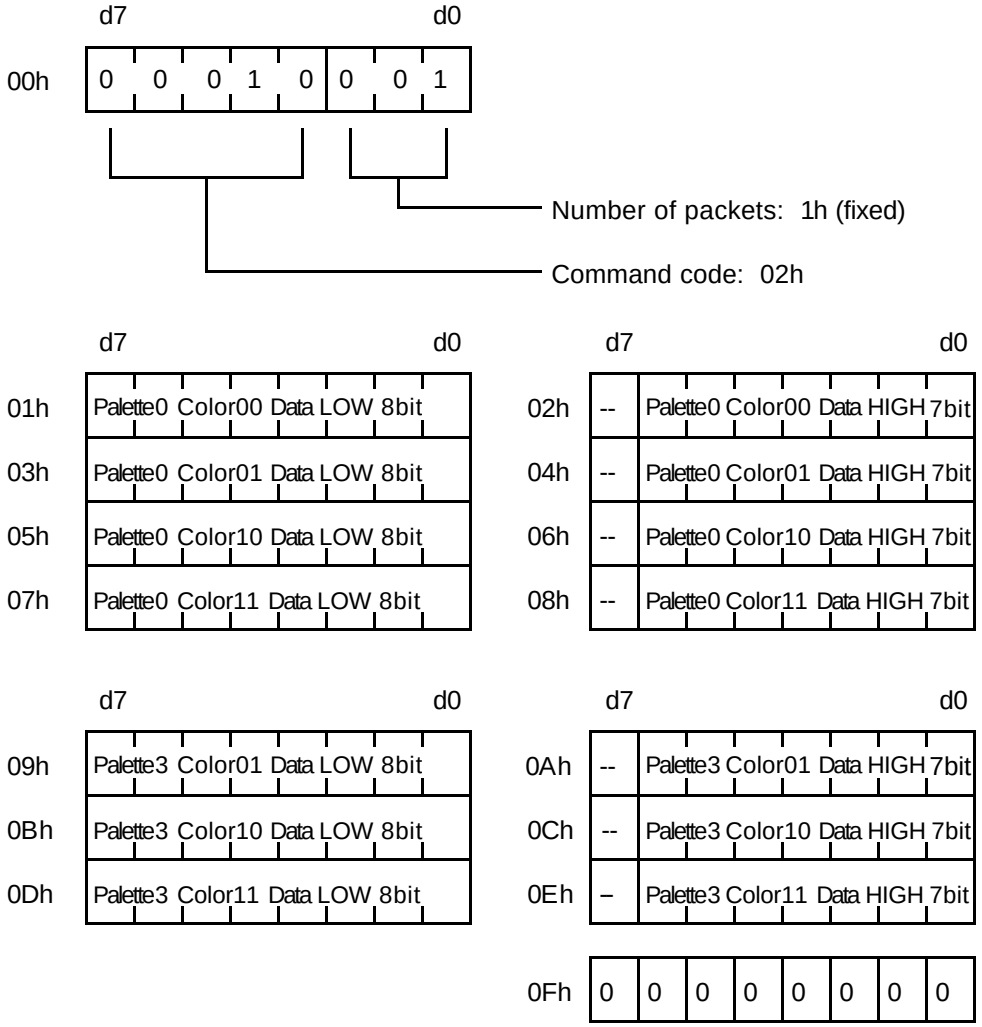
Command: PAL23 (Code: 01h)

Function: Sets the color data for SGB color palettes 2 and 3.


d7
d0
00h
0 0 0 0 1 0 0 1
Number of packets: 1h (fixed)
Command code: 01h
01h Palette2 Color00 Data LOW 8bit
02h -- Palette2 Color00 Data HIGH 7bit
03h Palette2 Color01 Data LOW 8bit
04h -- Palette2 Color01 Data HIGH 7bit
05h Palette2 Color10 Data LOW 8bit
06h -- Palette2 Color10 Data HIGH 7bit
07h Palette2 Color11 Data LOW 8bit
08h -- Palette2 Color11 Data HIGH 7bit
09h Palette3 Color01 Data LOW 8bit
0Ah -- Palette3 Color01 Data HIGH 7bit
0Bh Palette3 Color10 Data LOW 8bit
0Ch -- Palette3 Color10 Data HIGH 7bit
0Dh Palette3 Color11 Data LOW 8bit
0Eh -- Palette3 Color11 Data HIGH 7bit
0Fh 0 0 0 0 0 0 0 0

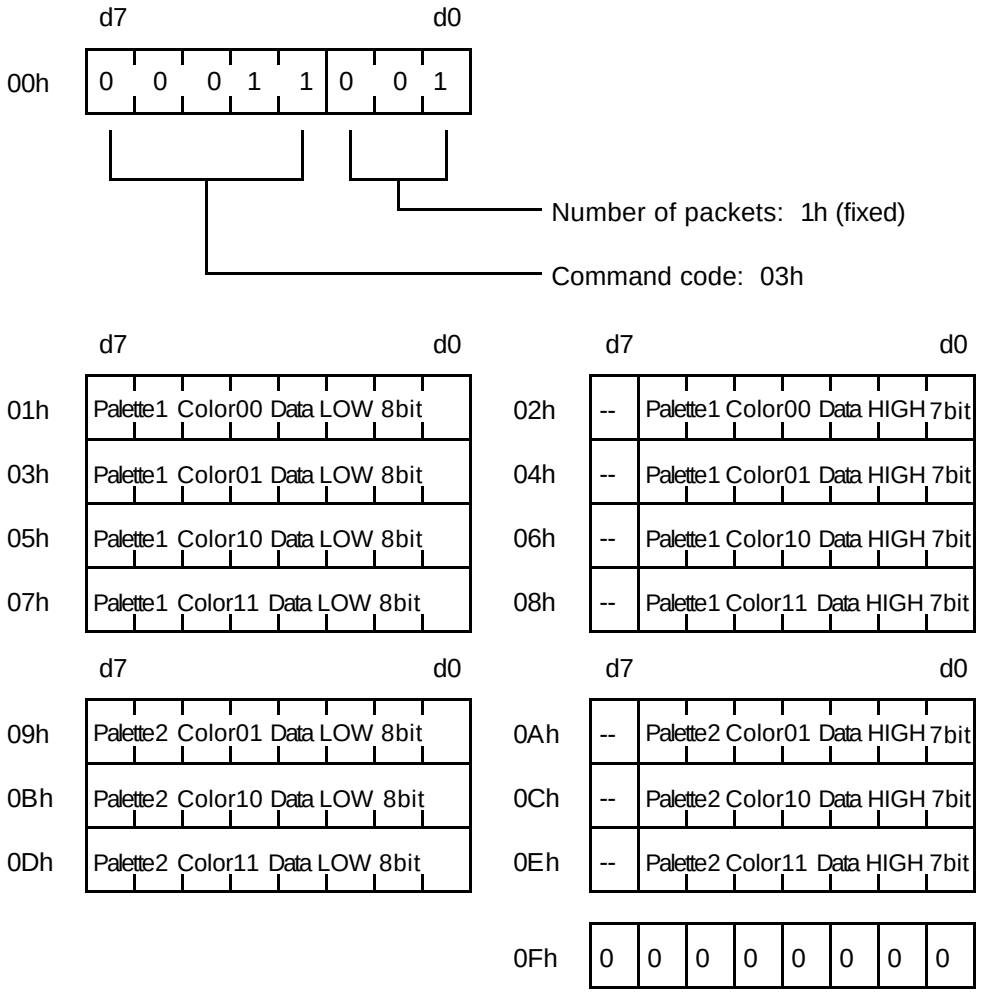
Command: PAL03 (Code: 02h)

Function: Sets the color data for SGB color palettes 0 and 3.


d7
d0
00h
0 0 0 1 0 0 0 1
Number of packets: 1h (fixed)
Command code: 02h
01h Palette0 Color00 Data LOW 8bit
02h -- Palette0 Color00 Data HIGH 7bit
03h Palette0 Color01 Data LOW 8bit
04h -- Palette0 Color01 Data HIGH 7bit
05h Palette0 Color10 Data LOW 8bit
06h -- Palette0 Color10 Data HIGH 7bit
07h Palette0 Color11 Data LOW 8bit
08h -- Palette0 Color11 Data HIGH 7bit
09h Palette3 Color01 Data LOW 8bit
0Ah -- Palette3 Color01 Data HIGH 7bit
0Bh Palette3 Color10 Data LOW 8bit
0Ch -- Palette3 Color10 Data HIGH 7bit
0Dh Palette3 Color11 Data LOW 8bit
0Eh – Palette3 Color11 Data HIGH 7bit
0Fh 0 0 0 0 0 0 0 0

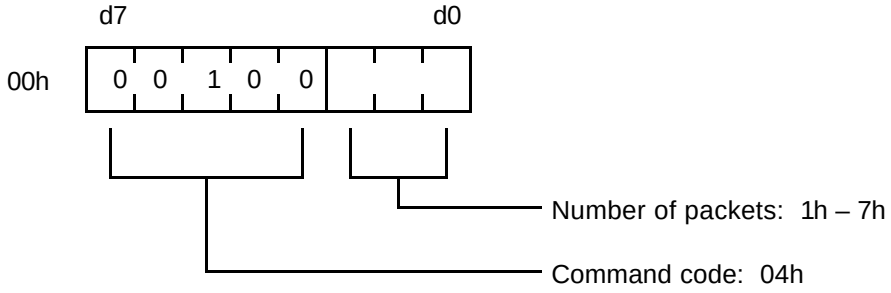
Command: PAL12 Code: 03h

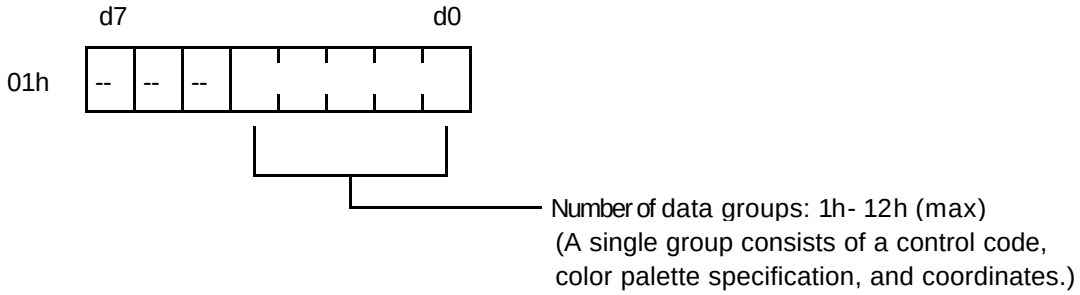
Function: Sets the color data for SGB color palettes 1 and 2.


d7
d0
00h
0 0 0 1 1 0 0 1
Number of packets: 1h (fixed)
Command code: 03h
01h Palette1 Color00 Data LOW 8bit
02h -- Palette1 Color00 Data HIGH 7bit
03h Palette1 Color01 Data LOW 8bit
04h -- Palette1 Color01 Data HIGH 7bit
05h Palette1 Color10 Data LOW 8bit
06h -- Palette1 Color10 Data HIGH 7bit
07h Palette1 Color11 Data LOW 8bit
08h -- Palette1 Color11 Data HIGH 7bit
09h Palette2 Color01 Data LOW 8bit
0Ah -- Palette2 Color01 Data HIGH 7bit
0Bh Palette2 Color10 Data LOW 8bit
0Ch -- Palette2 Color10 Data HIGH 7bit
0Dh Palette2 Color11 Data LOW 8bit
0Eh -- Palette2 Color11 Data HIGH 7bit
0Fh 0 0 0 0 0 0 0 0

Command Code: ATTR_BLK (Code: 04h)

Function: Applies the specified color palette attributes to areas inside and outside the square.



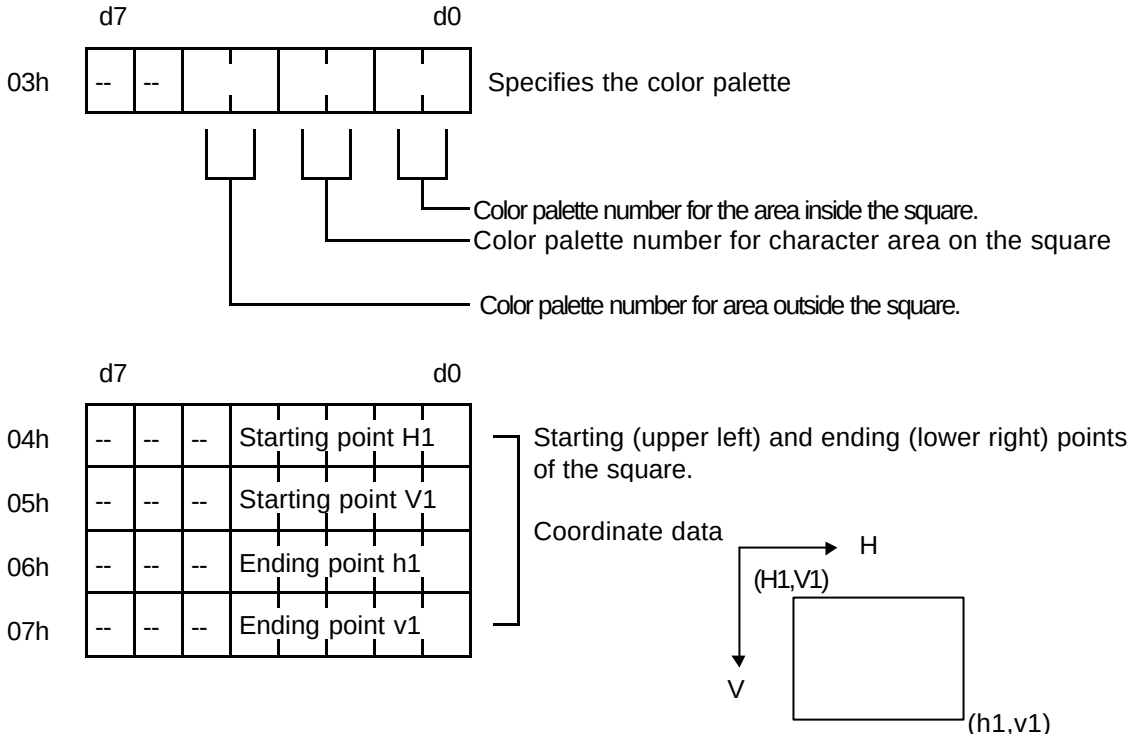


Control Code
Controls the attribute area according to the data in 03h.

Control Codes

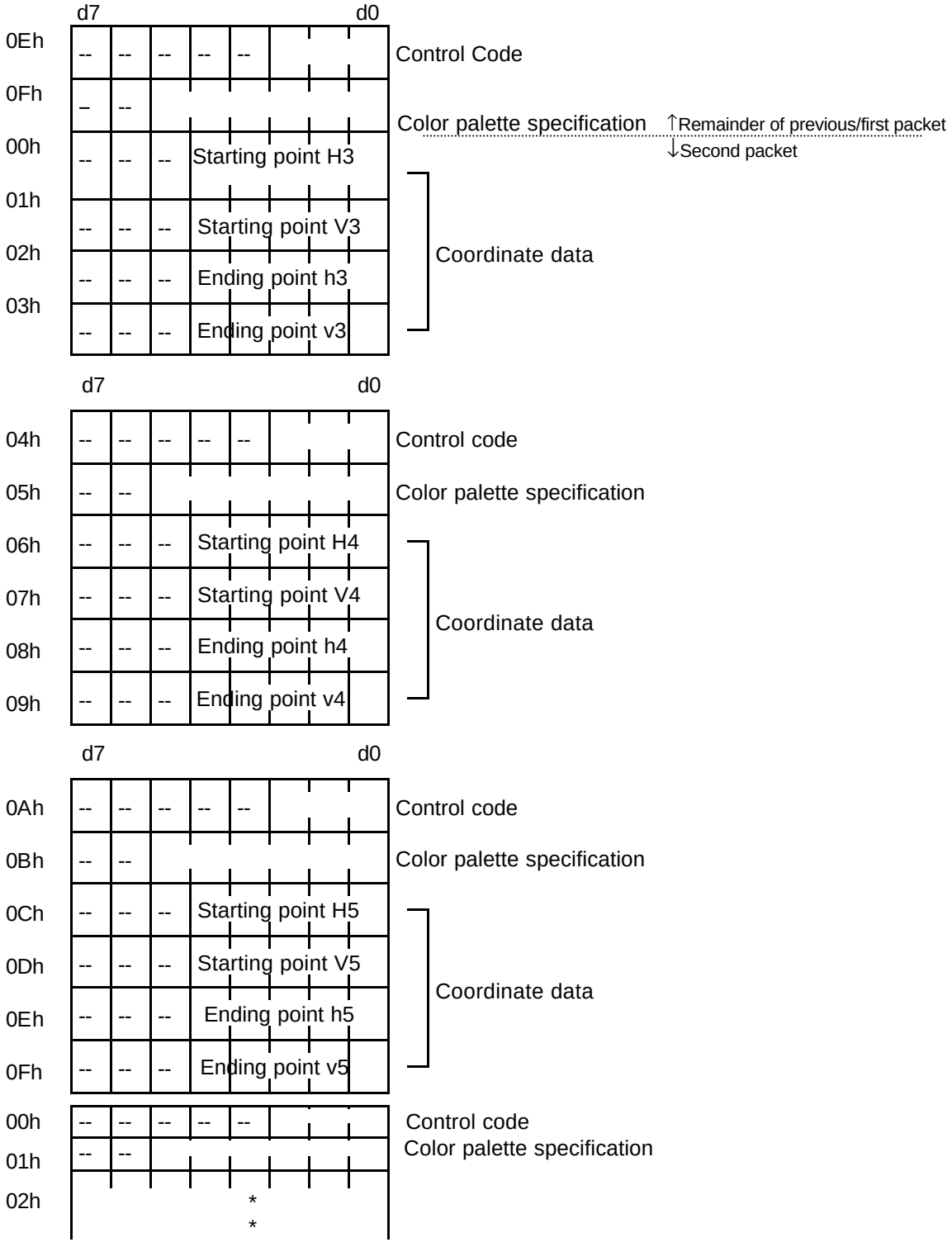
000	No control occurs.
001	Applies the attributes specified by d1 and d0 of 03h only to the area within the square (including the CHR border).
010	Applies the color palette attributes specified by d3 and d2 of 03h only on the square CHR border.
011	Applies the color palette attributes specified by d1 and d0 of 03h only to the area within the square, and applies the color palette attributes specified by d3 and d2 of 03h only to the border of the square.
100	Applies the attributes specified by d5 and d4 of 03h only to the area outside the square (including the CHR border).
101	Applies the color palette attributes specified by d1 and d0 of 03h to the area within the square, and applies the color palette attributes specified by d5 and d4 of 03h to the area outside of the CHR border. (CHR border is unchanged.)
110	Applies the color palette attributes specified by d5 and d4 of 03h only to the area outside of the square, and applies the color palette attributes specified by d3 and d2 of 03h to the CHR border .
111	Applies the specified color palette attributes to the area inside the square, to the CHR border line, and to the area outside the CHR border .

The color palette attributes of areas not specified are not changed.


d7
d0
03h
-- --
Specifies the color palette
Color palette number for the area inside the square.
Color palette number for character area on the square
Color palette number for area outside the square.
d7
d0
04h -- -- -- Starting point H1
05h -- -- -- Starting point V1
06h -- -- -- Ending point h1
07h -- -- -- Ending point v1
Starting (upper left) and ending (lower right) points of the square.
Coordinate data
H
(H1,V1)
V
(h1,v1)
d7
d0
08h -- -- -- -- --
Control Code
09h -- --
Specifies the color palette
0Ah -- -- -- Starting point H2
0Bh -- -- -- Starting point V2
0Ch -- -- -- Ending point h2
0Dh -- -- -- Ending point v2
Coordinate data
0Eh 0 0 0 0 0 0 0 0
0Fh 0 0 0 0 0 0 0 0

Note:	***If the number of packets is 1, 00h is written to 0Eh and 0Fh. If the number of packets exceeds 1, the control code and color palette specification code of the next data item are written to 0Eh and 0Fh, respectively.***

When the number of packets exceeds 1:



* The empty area of the packet is filled with 00h.

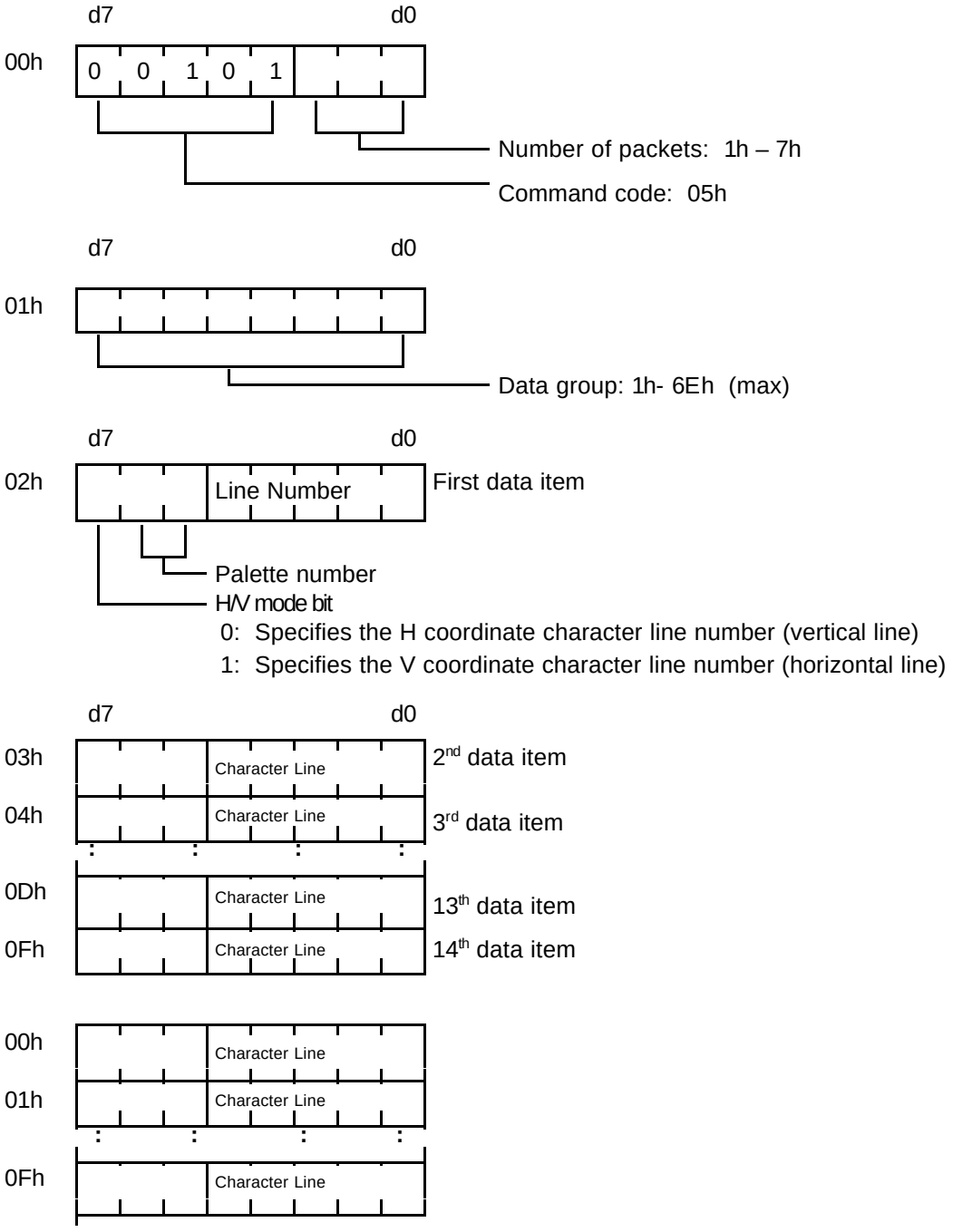
Note:	***When there is no area inside the square border (e.g., h1 = H1 + 1), a control code such as one that sets the color attribute for the area inside the border cannot be used.***

Please note that when ATTR_BLK, ATTR_LIN, ATTR_DIV, or ATTR_CHR are used, the data that is sent are valid even if MASK_EN (freezes screen immediately before masking) is selected.

When using MASK_EN before these commands, use 10h or 11h as the argument. If 01h is used as the MASK_EN argument, ATTR_TRN and ATTR_SET should be used.

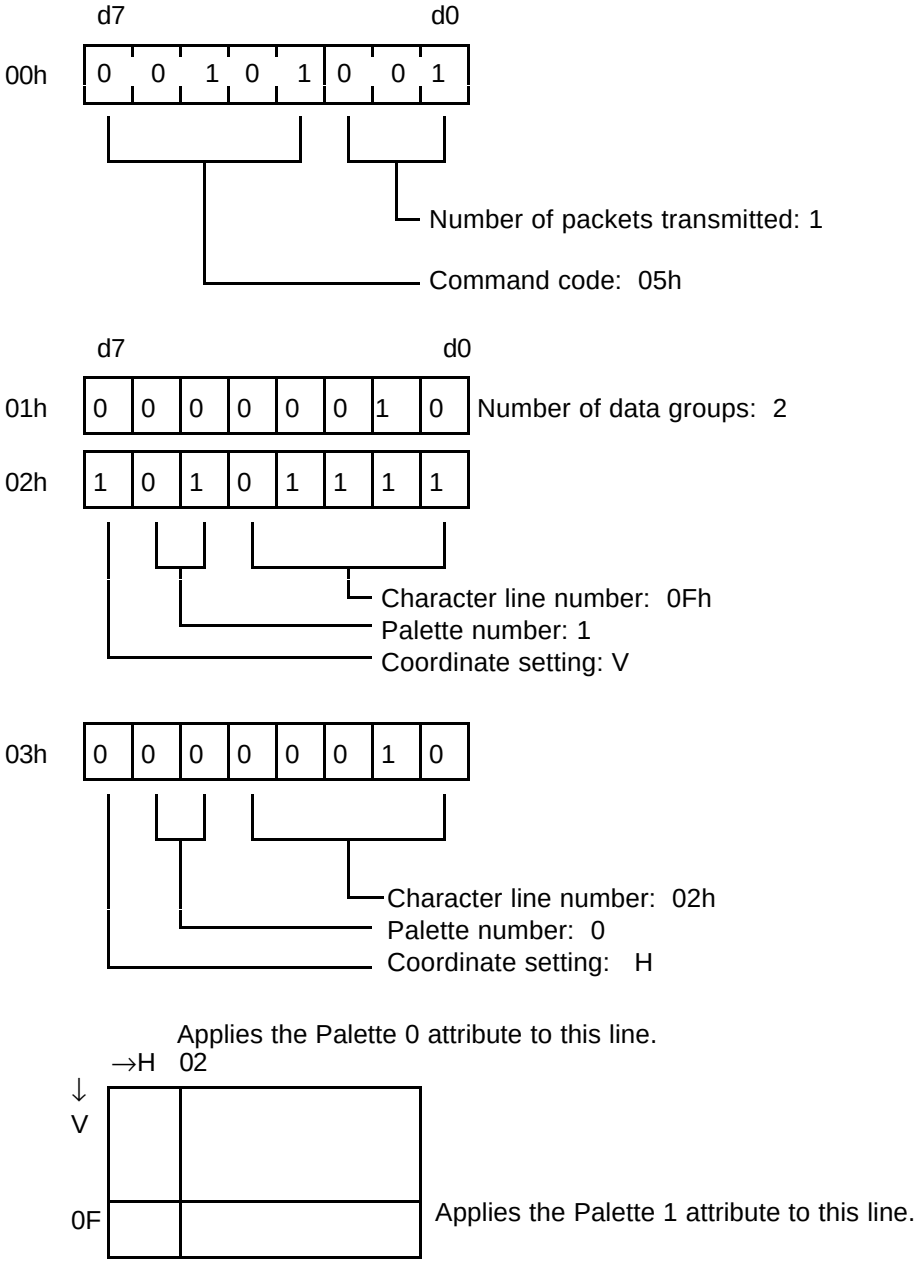
Command: ATTR_LIN (Code: 05h)

Function: Applies the specified color palette attribute to a coordinate line.



* See the note on ATTR_BLK.

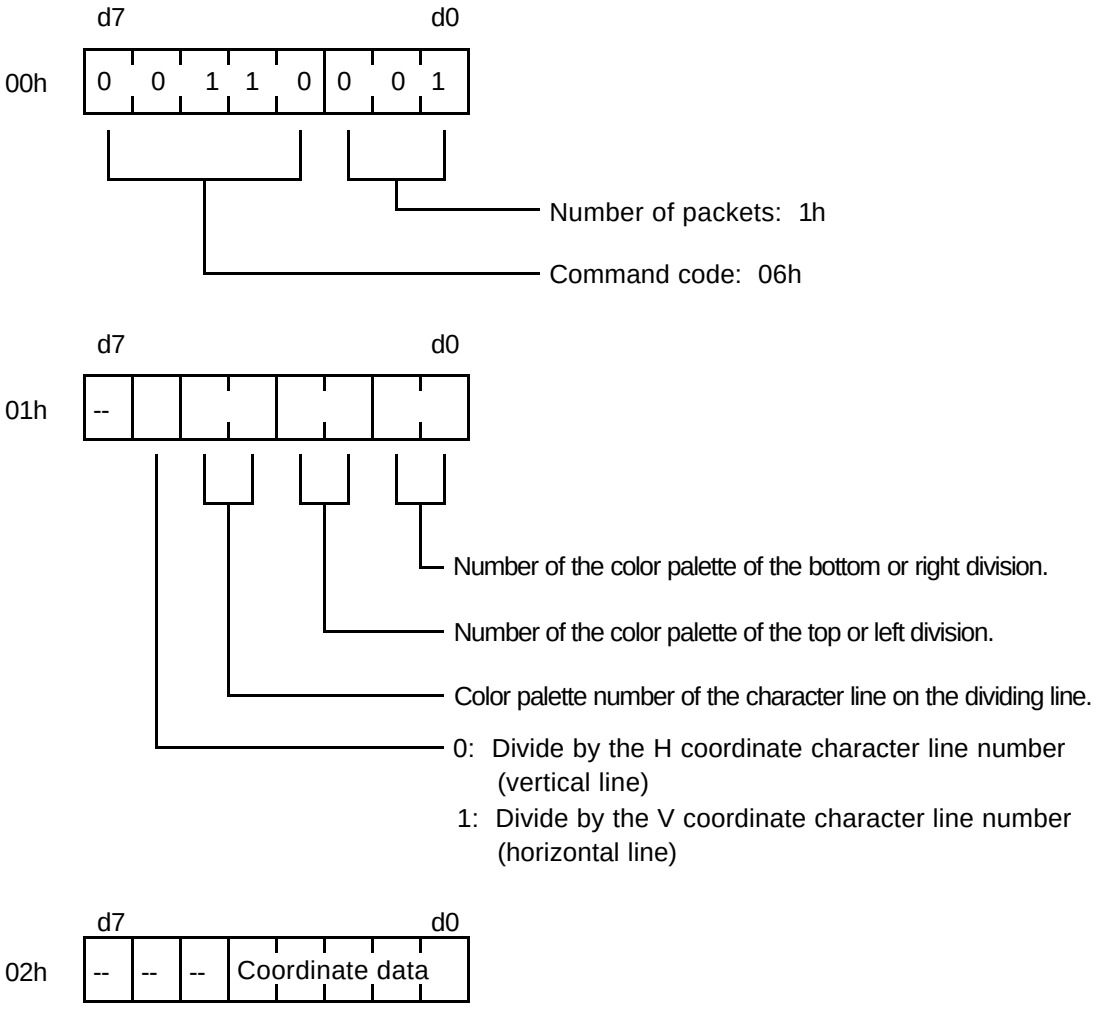
Example:



* The color of intersection of the two lines is decided by the last line color.

Command: ATTR_DIV (Code: 06h)

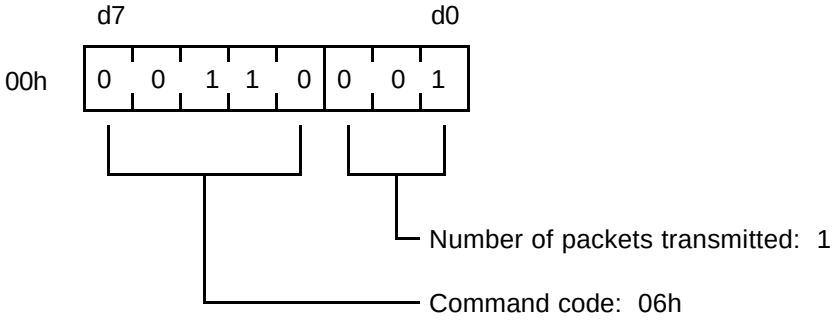
Function: Divides the color palette attributes of the screen by the specified coordinates.

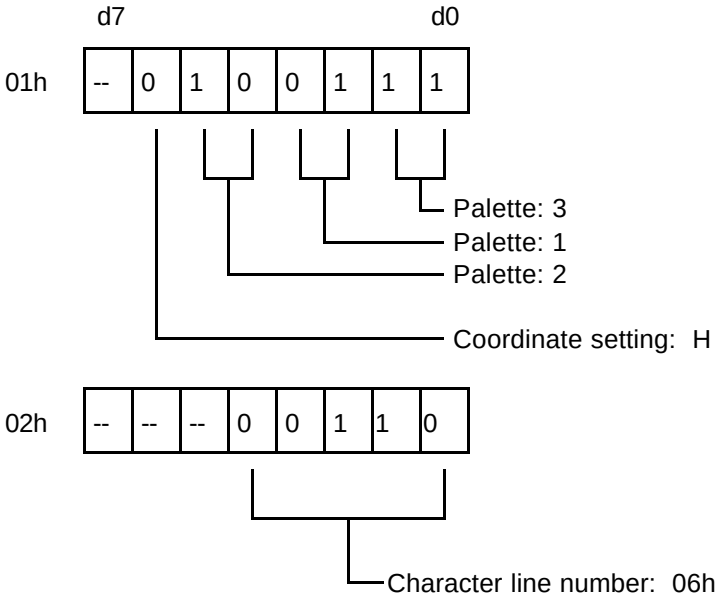

d7
d0
00h
0 0 1 1 0 0 0 1
Number of packets: 1h
Command code: 06h
d7
d0
01h
--
Number of the color palette of the bottom or right division.
Number of the color palette of the top or left division.
Color palette number of the character line on the dividing line.
0: Divide by the H coordinate character line number (vertical line)
1: Divide by the V coordinate character line number (horizontal line)
d7
d0
02h
-- -- -- Coordinate data

* 03h - 0Fh should be filled with 00h.

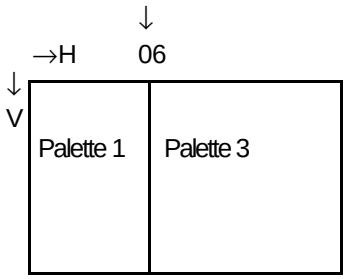
* See note on ATTR_BLK.

Example:



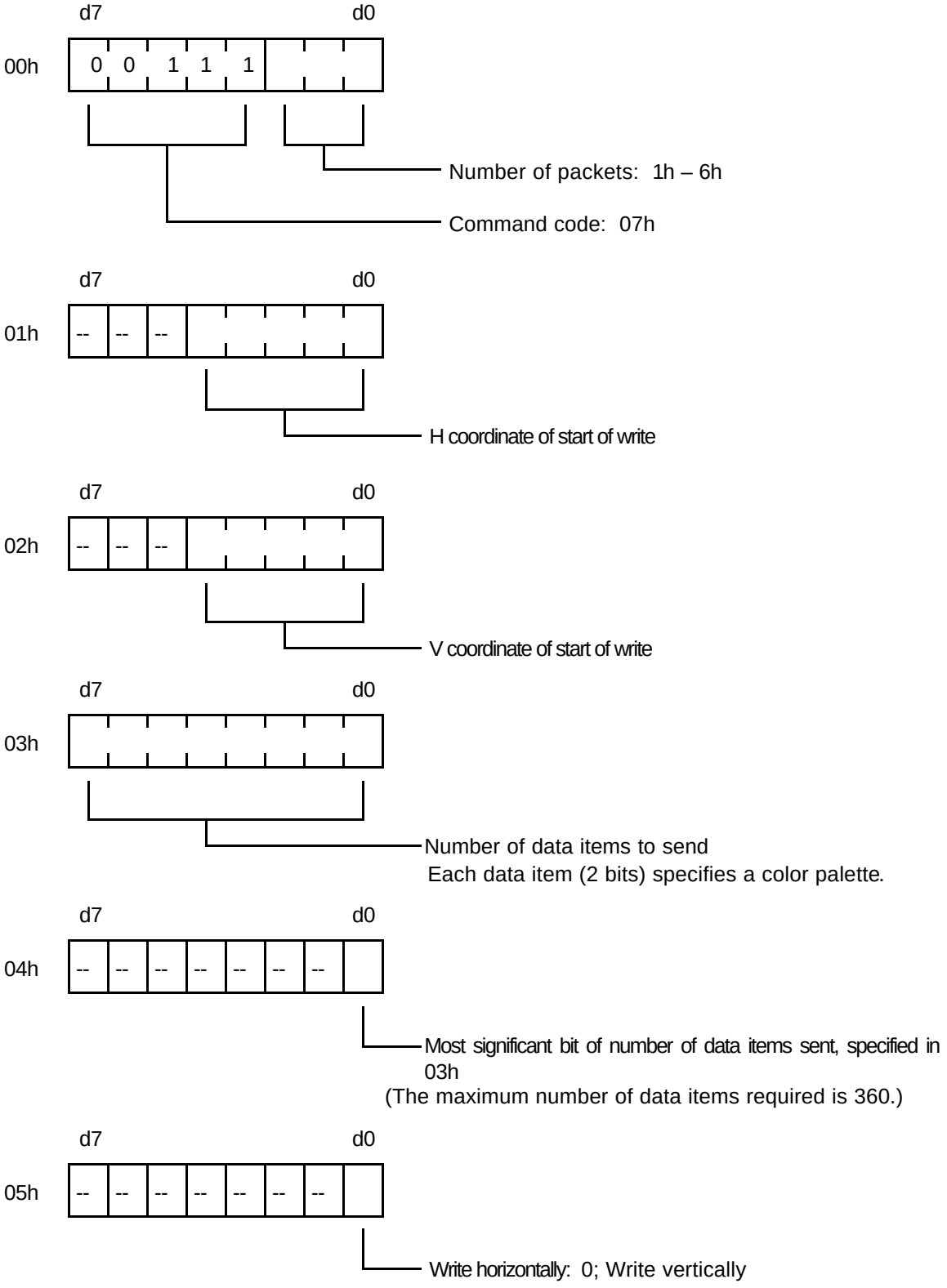


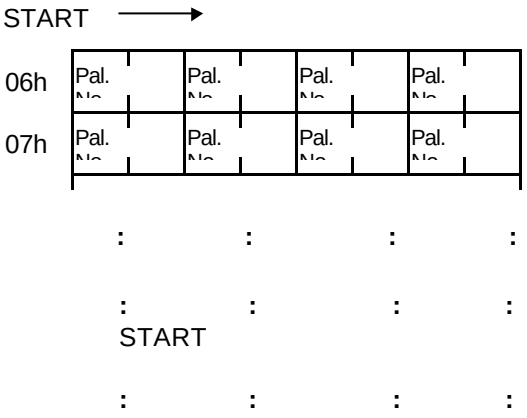
Sets this character line to the Palette 2 attribute.



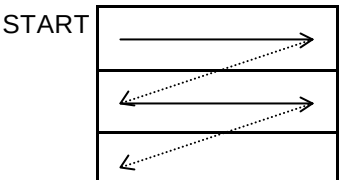
Command: ATTRIBUTE_CHR (Code: 07h)

Function: Specifies a color palette for each character.

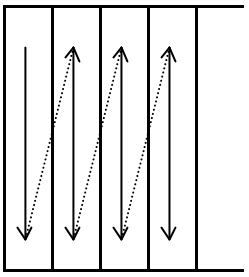




Horizontal write (H direction)



Vertical write (V direction)



: : : :

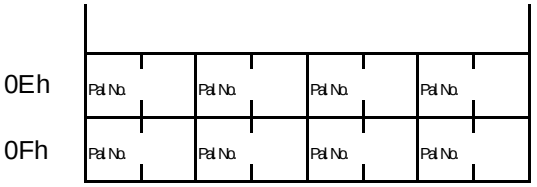
: : : :
START

: : : :

Sending color palette data for entire screen:

: : : :

6th Packet

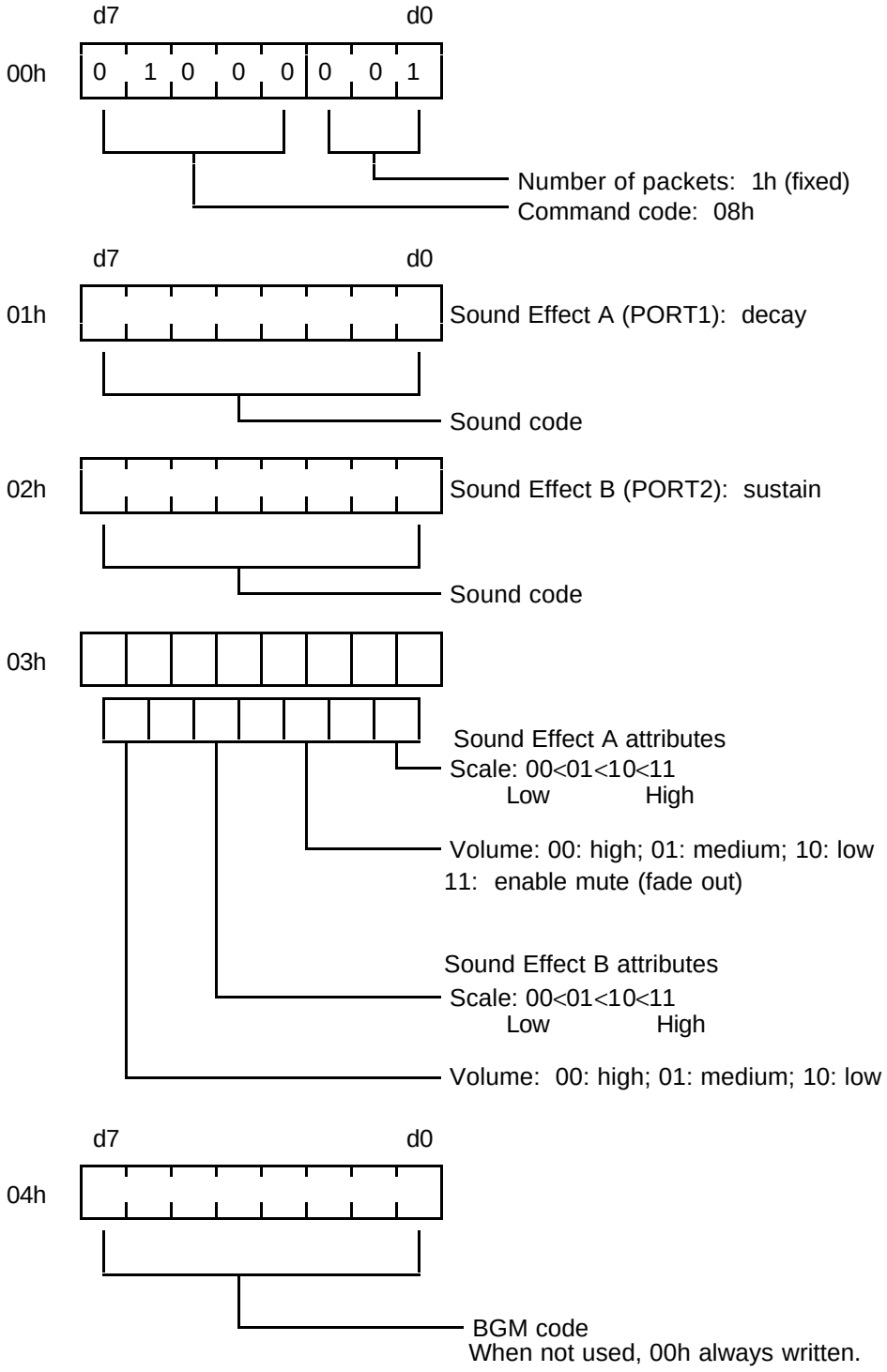


← Data items nos. 357, 358, 359, and 360.

* See note on ATTR_BLK.

Command: SOUND (Code: 08h)

Function: Generates and halts internal sound effects and sounds that use internal tone data.

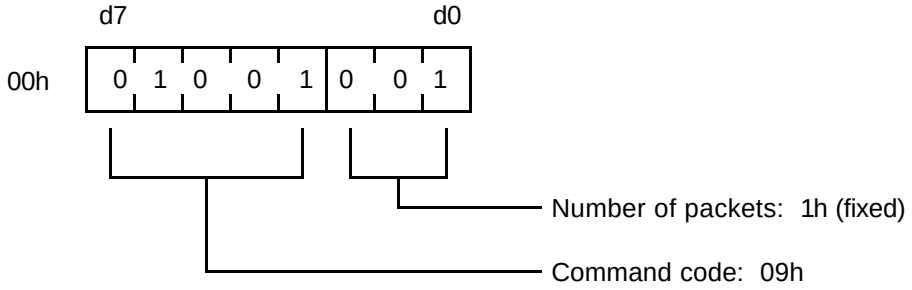


* For more information, see Section 3.4, Sound Flag Lists.

Command: SOU_TRN (Code: 09h)

(Data transfer using VRAM)

Function: Sends a sound program and sound data to the APU.



* The 4 Kbytes of SGB RAM data immediately following the command is sent to APU RAM.

The data to be transferred must be prepared prior to the frame preceding issuance of the command.

The transfer ends 6 frames after the command is issued (not counting the frame in which the command is issued).

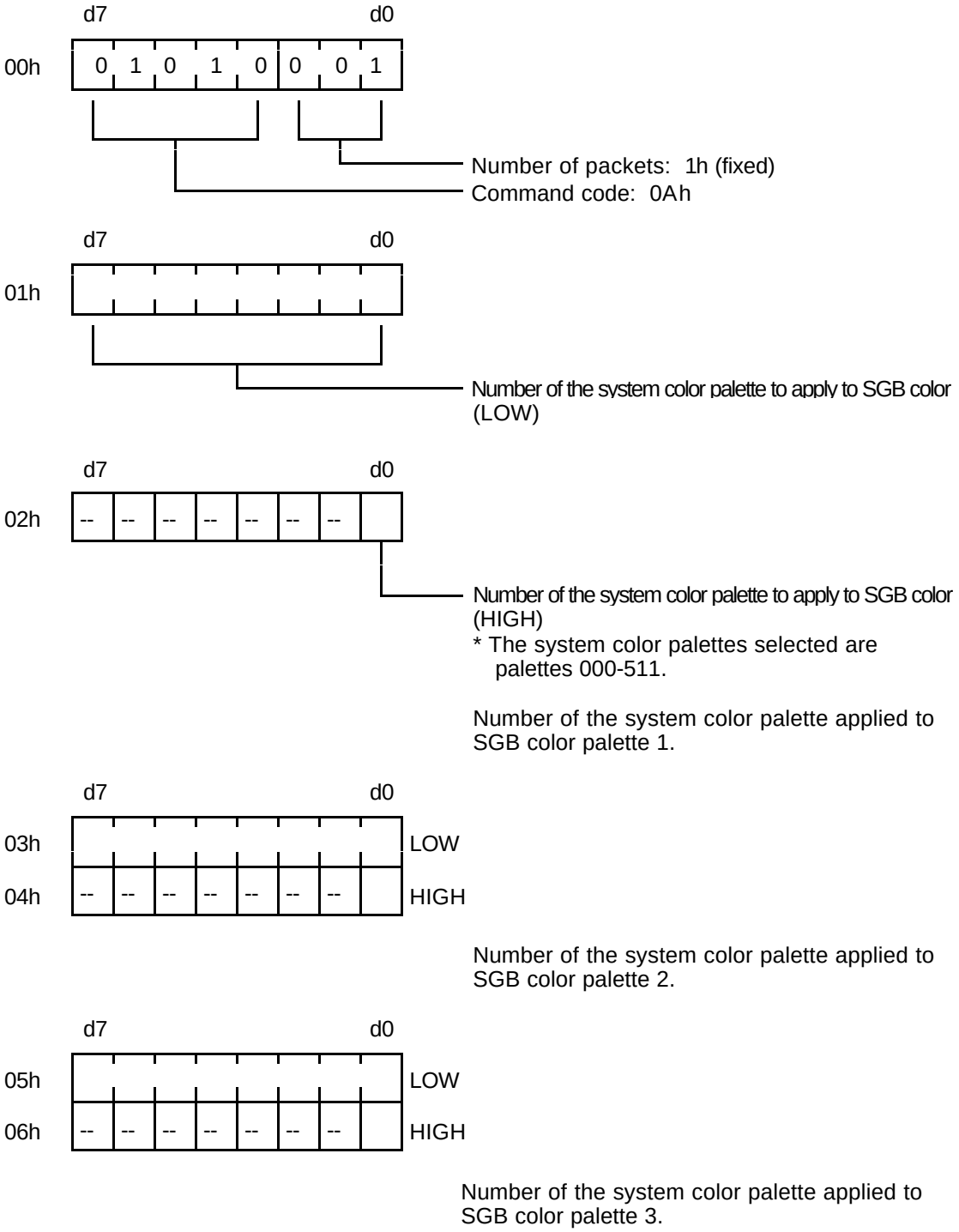
The beginning of the data for transfer contains a 16-bit representation of the number of data items and the transfer destination address, and the end contains an ending code and the starting address of the program. For more information, see Chapter 7: *Super Game Boy Sound*.

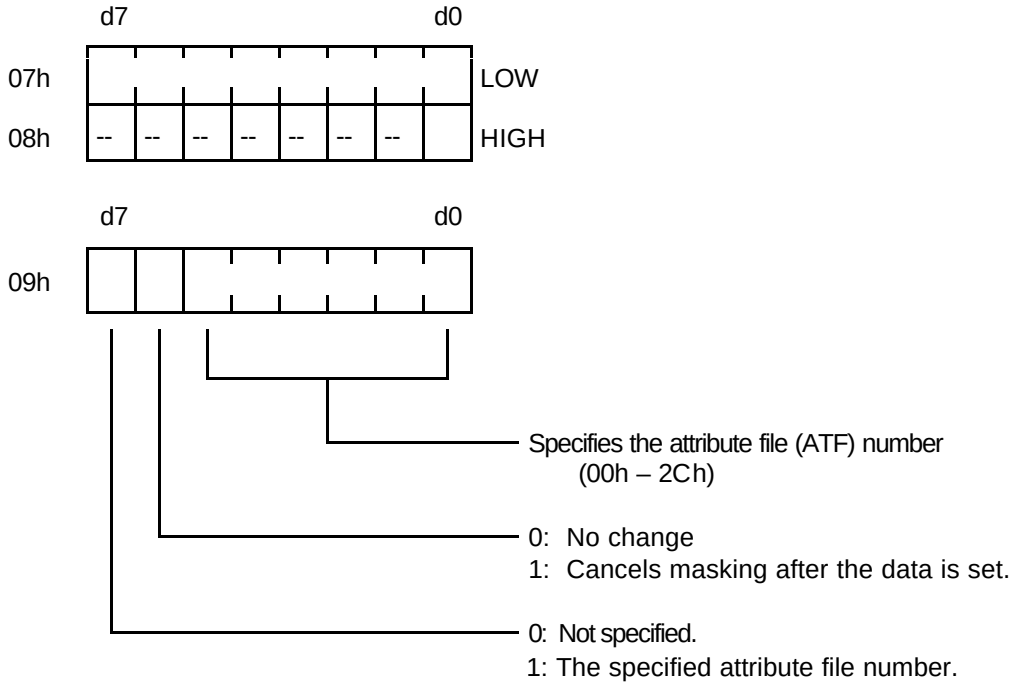
APU RAM program area: 0400h – 2AFFh/9.75 Kbytes
APU RAM music data area: 2B00h – 4AFFh/8 Kbytes
APU RAM sampling data area: 4DB0h-EEFFh/40.25 Kbytes

Note:	***When SOU_TRN is used, 5 packets of SOU_TRN initialization should be sent to the register file. For more information, see Section 6.5.3, SOU_TRN Initialization Data.***

Command: PAL_SET (Code: 0Ah)

Function: Applies system color palettes to SGB color palettes.



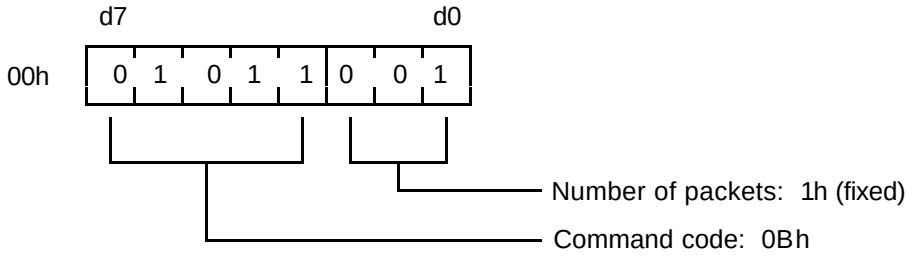


* 0Ah - 0Fh should be filled with 00h.

Command: PAL_TRN (Code: 0Bh)

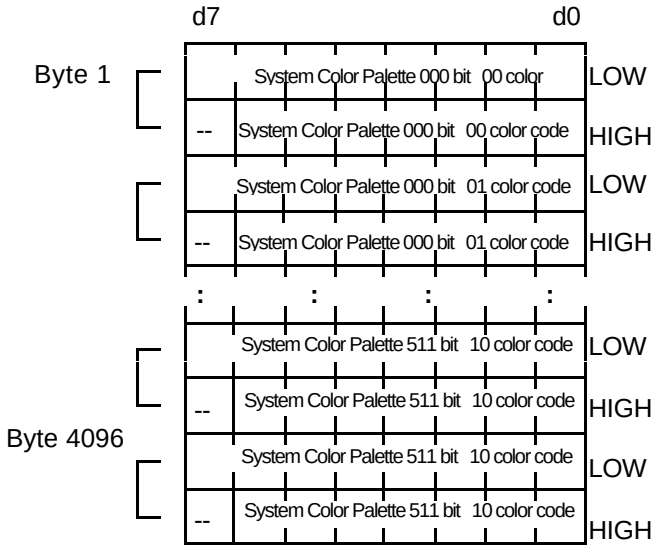
(Data Transfer using VRAM)

Function: Transfers color data to the system color palette.



* The 4 Kbytes of SGB RAM data immediate following the command is handled as system color palette data and stored in Super NES W-RAM as data for system color palettes 000 – 511.

The format of data storage in SGB RAM is as follows.

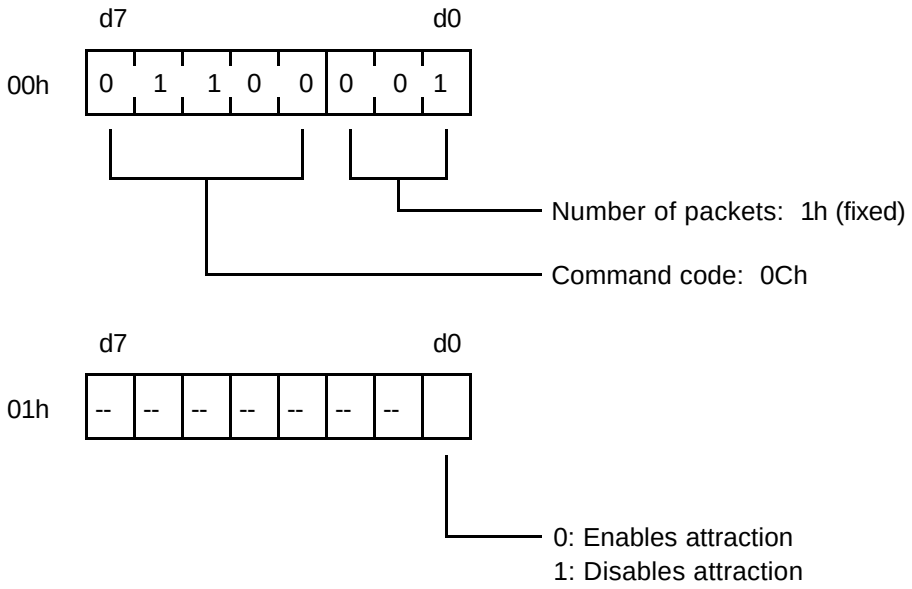


The storage addresses are 3000h - 3FFFh.

Command: ATRC_EN (Code: 0Ch)

Function: Enables and disables attraction mode.

Enables and disables attraction on the picture frame.
The default setting is enabled (00h).
If the command is issued during attraction, attraction is stopped.



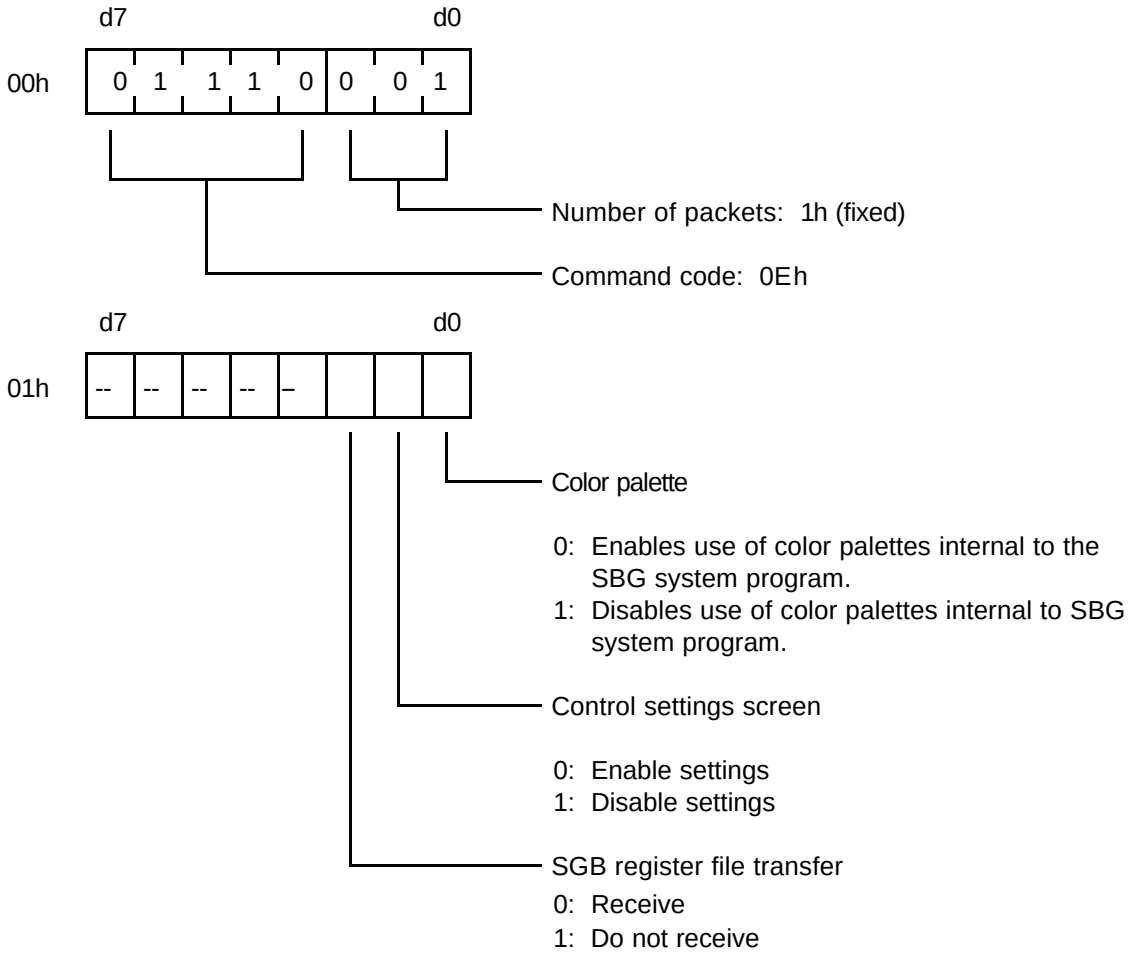
Example: Attraction start duration for a model (type without communication connector).

The time required for attraction to start for each picture frame is as follows. (Times shown in parentheses are times required to start attraction a second time.)

Mario	7 min. (5 min.)
Cork	3 min. (5 min.)
Landscape	1 min. (1 min.)
Cinema	3 mins.
Cats	3 mins.
Pencils	3 mins. (5 mins.)
Escher art	7 mins. (5 mins.)

Command: ICON_EN (Code: 0Eh)

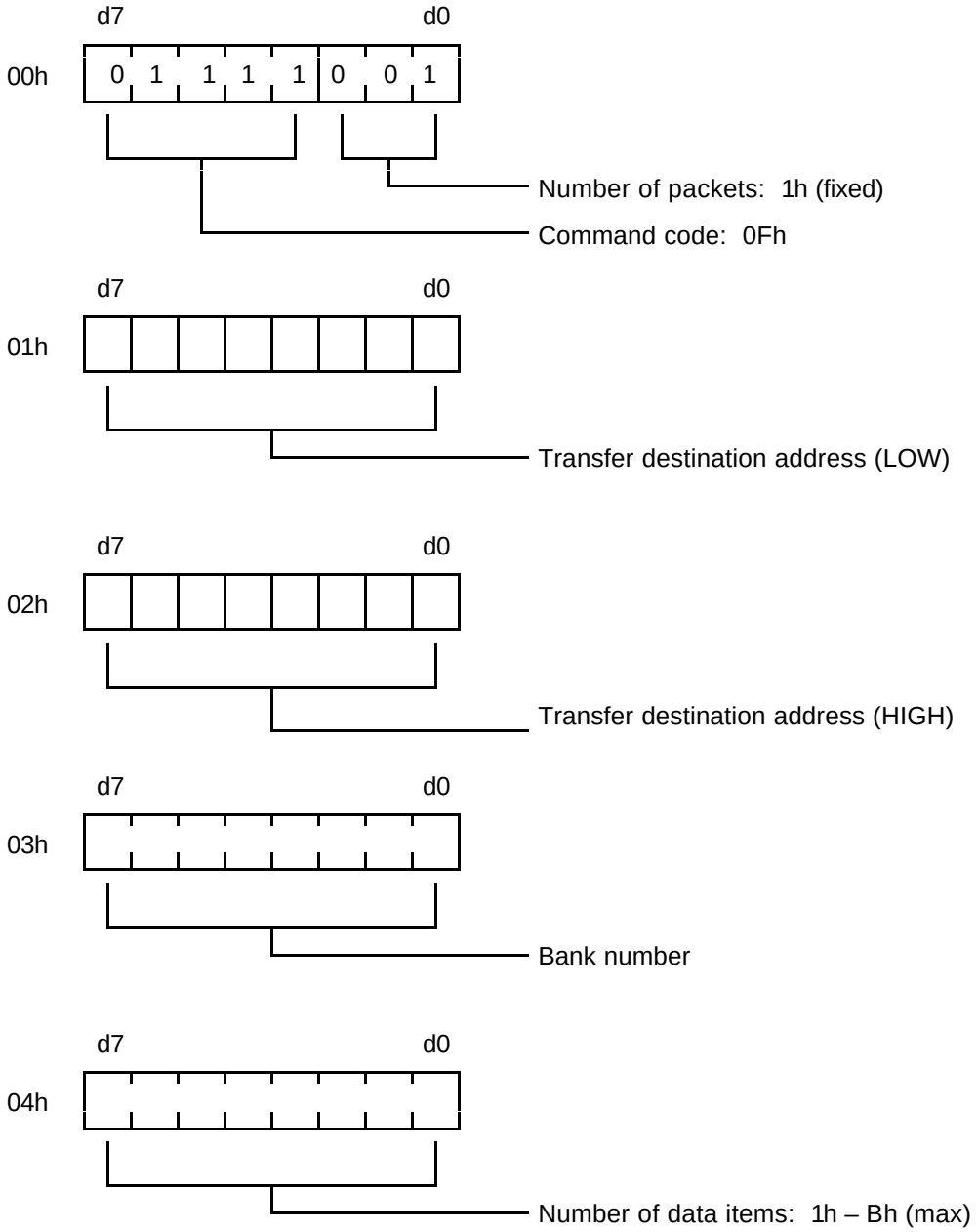
Function: Enables and disables the icon function.

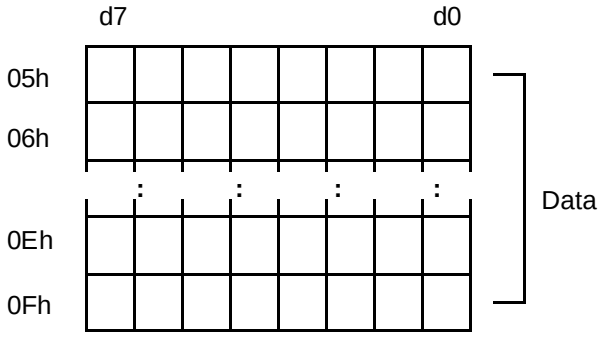


* The default value is 00h.

Command: DATA_SND (Code: 0Fh)

Function: Transfers data to Super NES W-RAM using the register file.





* Free Addresses

Bank 00h
1800h - 1FFFh

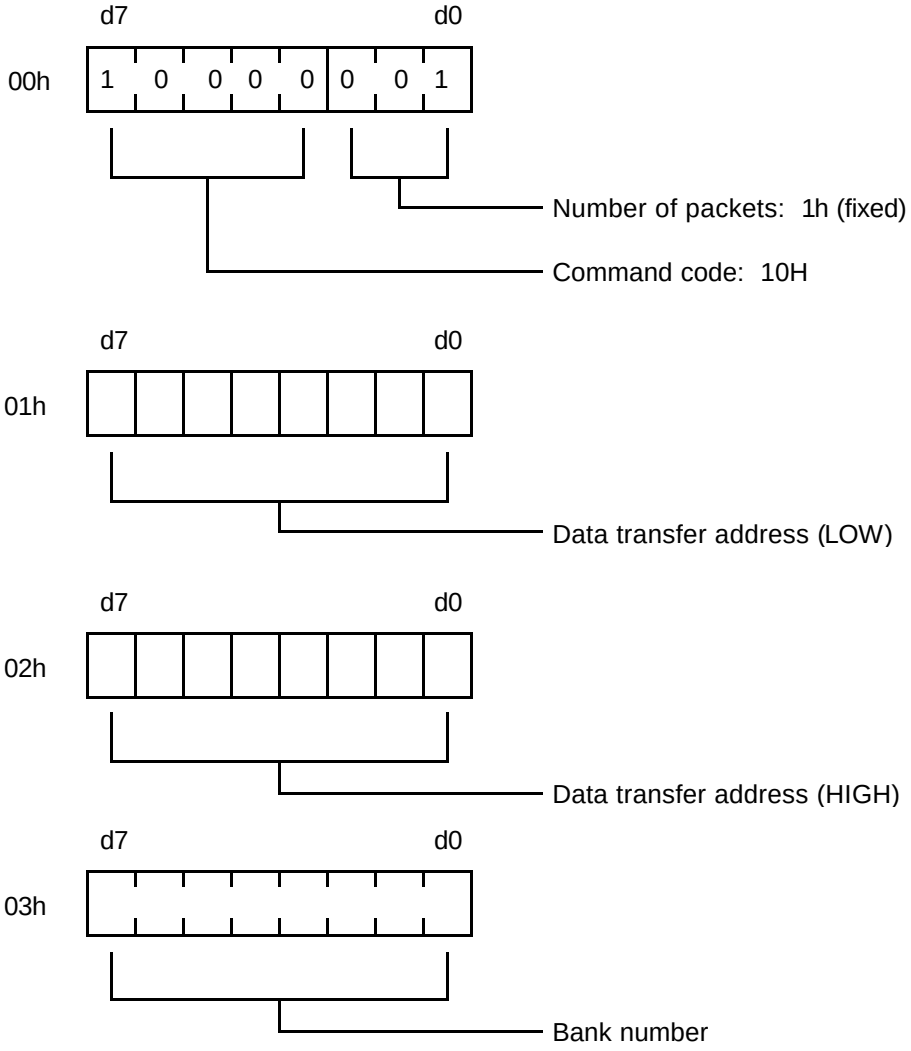
Bank 7Eh
B000h – BFFFh

Bank 7Fh
0000h – FFFFh

Command: DATA_TRN (Code: 10h)

(Data Transfer using VRAM)

Function: Transfers data in SGB RAM to Super NES W-RAM.



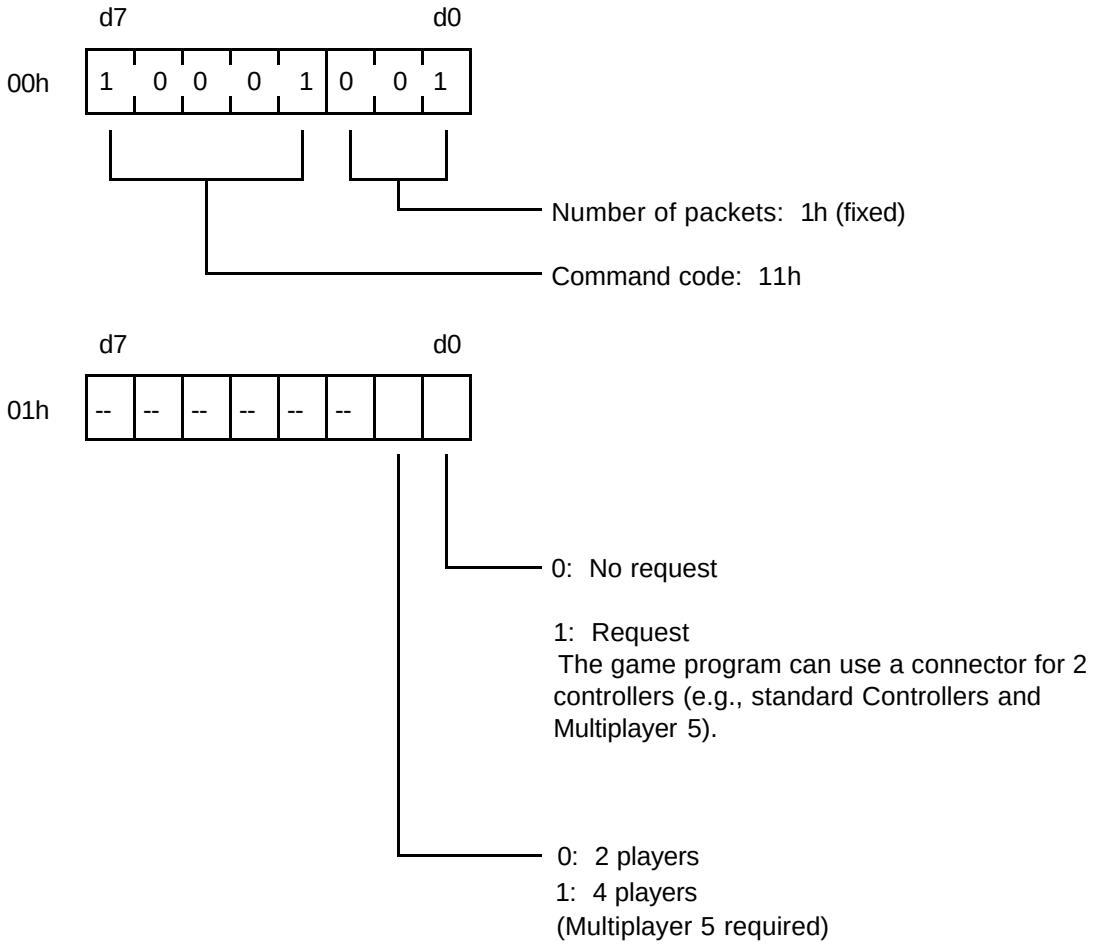
* Free Addresses

Bank 00h	1800h – 1FFFh
Bank 7Eh	B000h – BFFFh
Bank 7Fh	0000h – FFFFh

Note: ***When an Super NES program is tranferred to W-RAM and executed, 00h should be written to 1700h of bank 00. This can be written either by using DATA_SND or DATA_TRN or by using the transferred program.***

Command: MLT_REQ (Code: 11h)

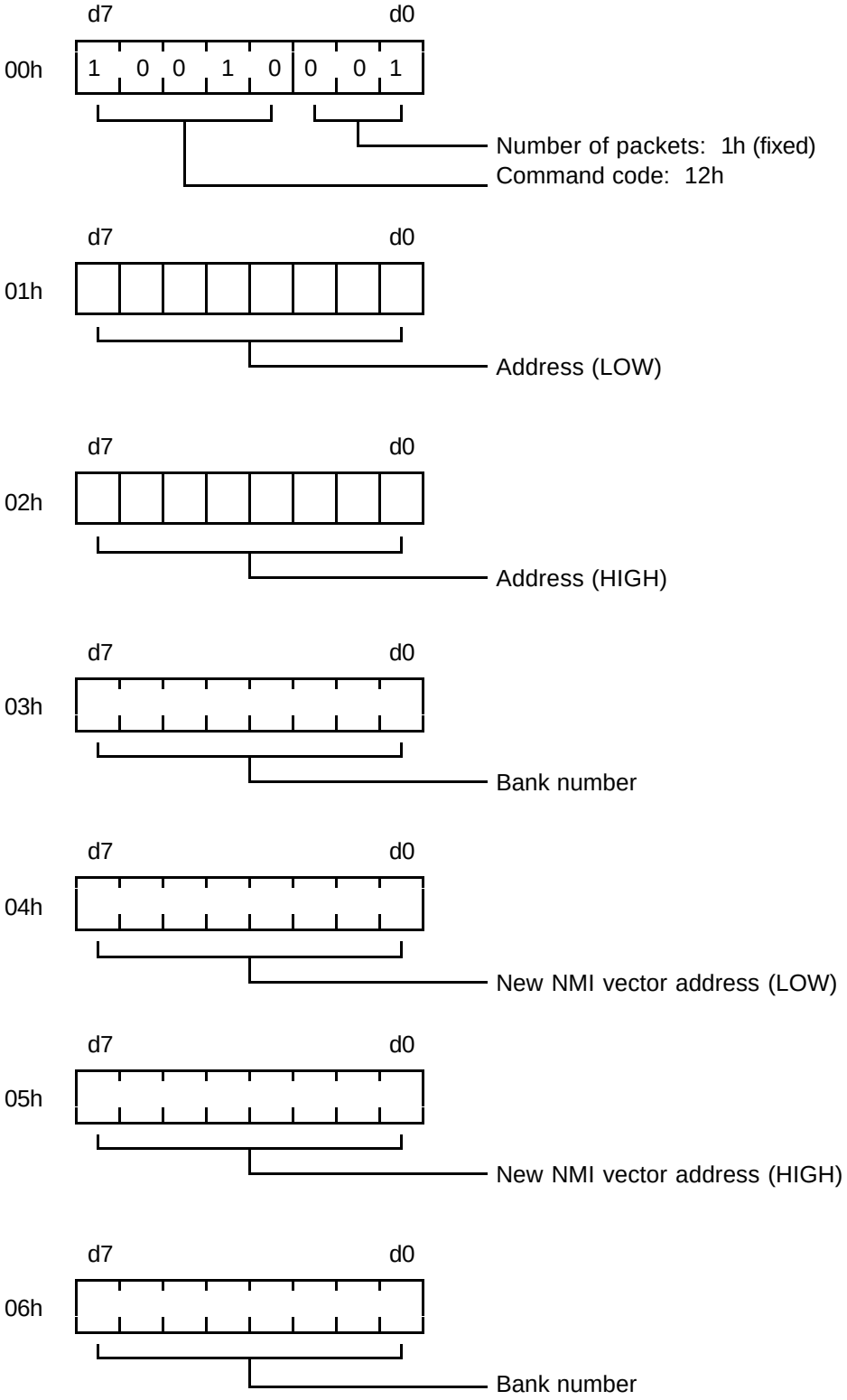
Function: Requests multiplayer mode.



* The default value is 00h.

Command: JUMP (Code: 12h)

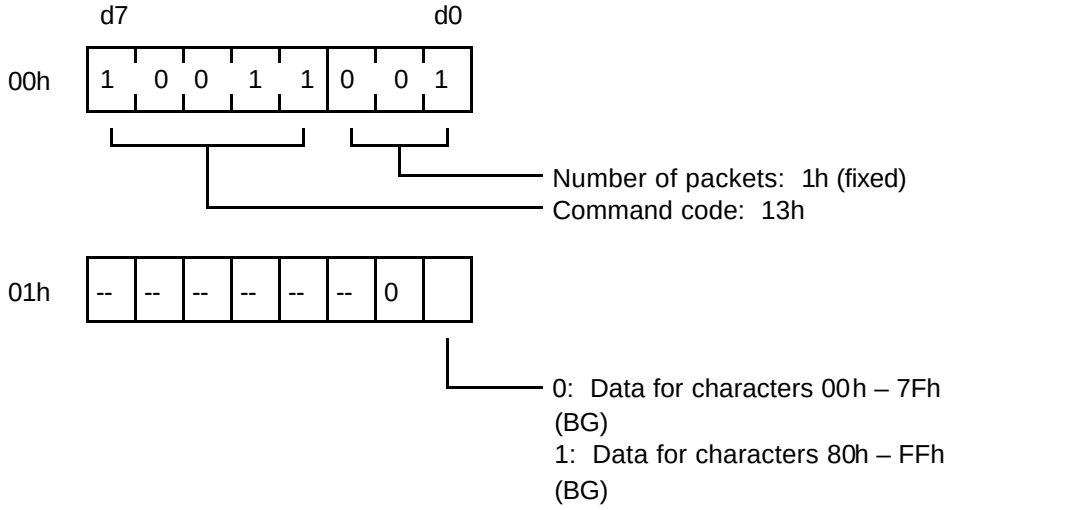
Function: Sets the Super NES program counter to the specified address.



Note:	***If all addresses from 04h to 06h are set to 0, the NMI jumps to the original vector. NMI is disabled in the system program, so it must be enabled to be used.***

Command: CHR_TRN (Code: 13h)

(Data Transfer using VRAM) Function: Transfers Super NES character format data.


d7
d0
00h
1 0 0 1 1 0 0 1
Number of packets: 1h (fixed)
Command code: 13h
01h
-- -- -- -- -- -- 0
0: Data for characters 00h – 7Fh (BG)
1: Data for characters 80h – FFh (BG)

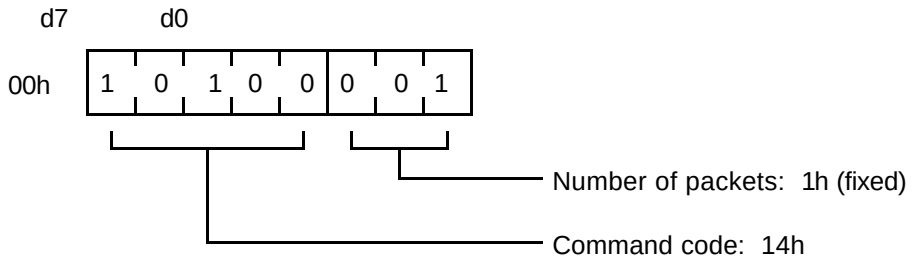
The characters are in 16-color (4-bit) mode.

Note:	***The 4 Kbytes of SGB RAM data immediately following this command is handled as Super NES character data and transferred to Super NES VRAM.***
	The format of the tranferred data is based on the Super NES character data format.
	The BG character names are allocated to 00h – FFh.
	When character data is used for the picture frame, characters with a character name setting of 0x00h should consist of null bits, and all dots of characters with a name setting of 0x01h should be represented by non-null bits.

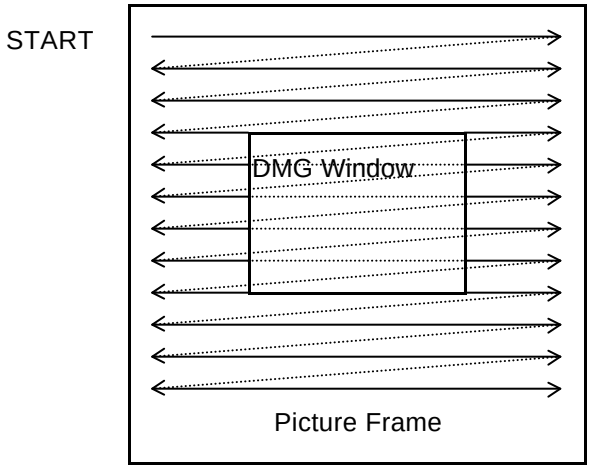
Command: PCT_TRN (Code: 14h)

(Data Transfer using VRAM)

Function: Transfers screen data and color data for picture frames created by the software developer.



* The 4 Kbytes of SGB RAM immediately following this command are handled as screen data and transferred to Super NES VRAM.



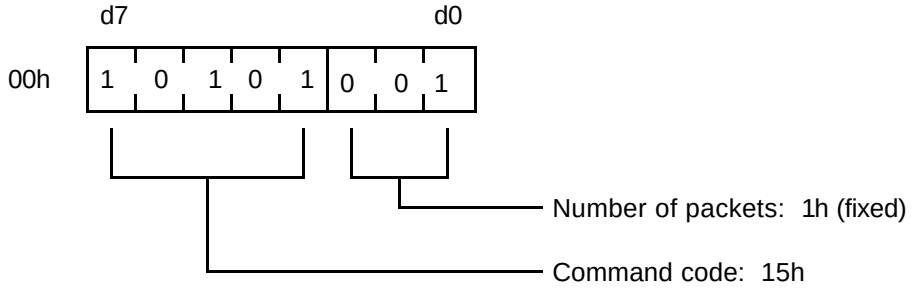
(Number of characters: 1,024)

There should be 1,024 uncompressed characters of screen data.
The inside of the DMG window should be filled with null characters.
Three color palettes, 4-6, are transferred.
The initial data consists of 2,048 bytes of screen data. This is followed by by 3 palettes of color data (2 bytes x 16 x 3).

The format of the transferred data is based on that of Super NES screen and color data. However, the BG priority bit is set to 0, the color palettes to palette numbers 4-6, the higher-order 2 bits of the character name to 00b, and the characters to 8 x 8-bit mode.

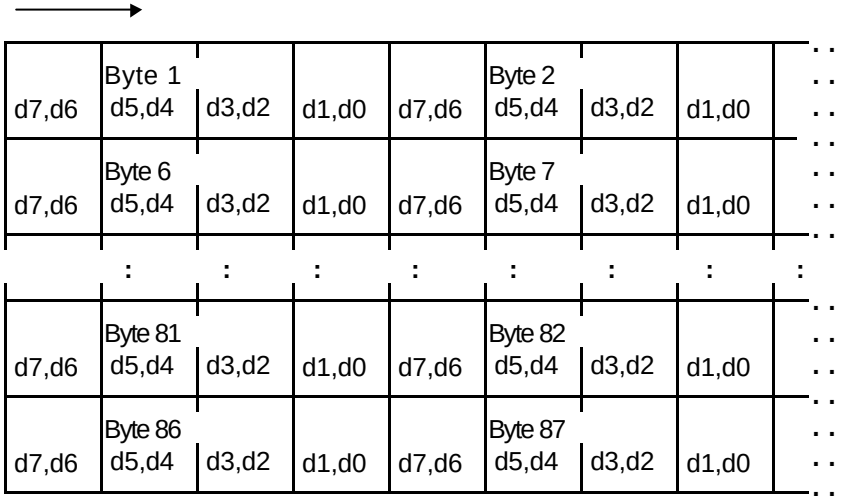
Command: ATTR_TRN (Code: 15h)

(Data Transfer using VRAM) Function: Transfers attribute files.



* The 4 Kbytes of SGB RAM immediately following this command are transferred to W-RAM as attribute files. (The capacity of each attribute file is 2 x 20 x 18/8 = 90 bytes. Thus, 45 attribute files occupy 4,050 bytes, ATF0h-ATF44h.

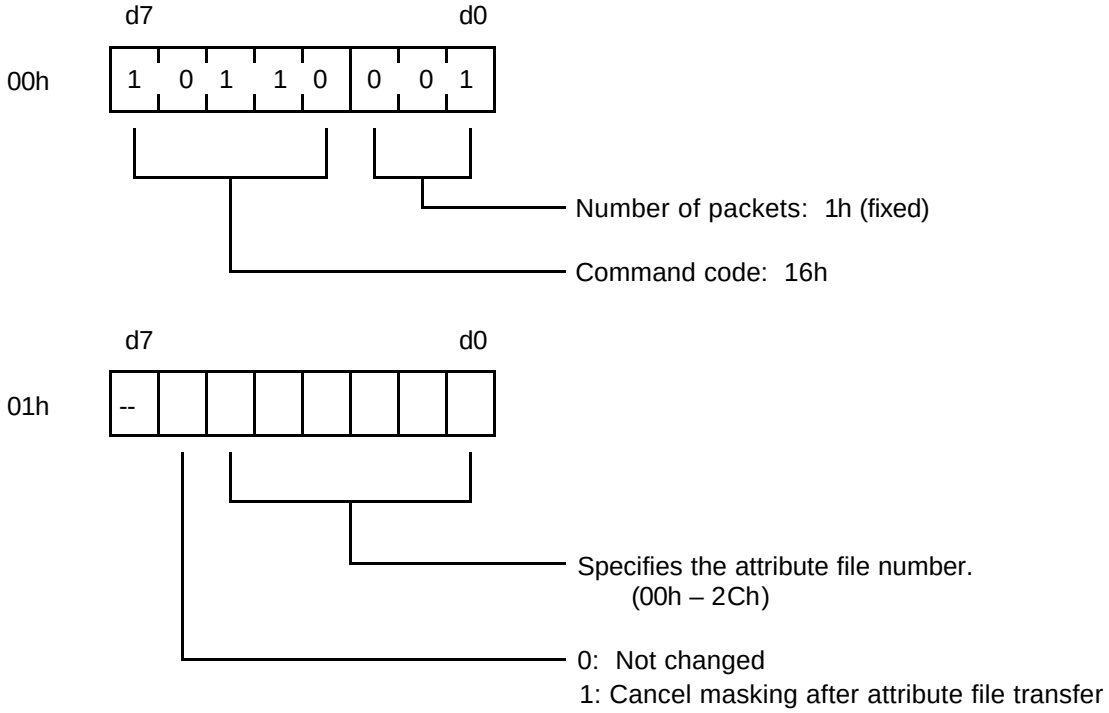
The ATF data format (90 bytes total) — written horizontally from the left edge of the DMG window.



(The figure depicts a DMG window with 20 x 18 characters.)

Command: ATTR_SET (Code: 16h)

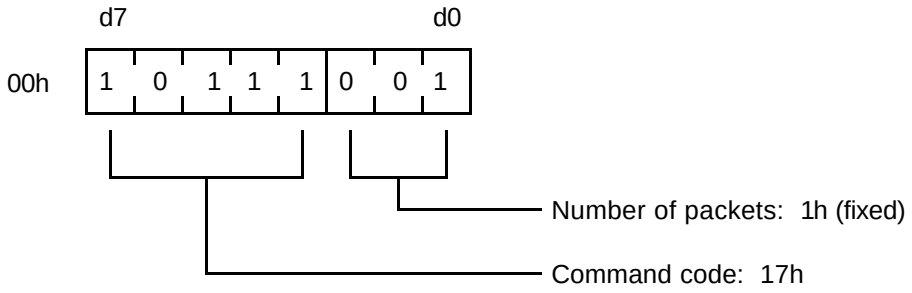
Function: Applies the specified attribute file to the DMG window.

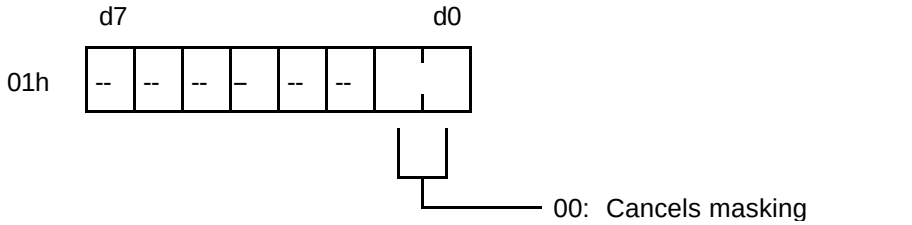


* 02h - 0Fh filled with 00h.

Command: MASK_EN (Code: 17h)

Function: Masks the DMG window.





00: Cancels masking
01: Freezes the screen immediately before masking. (No transfers to Super NES VRAM are performed from after the command is issued until masking is canceled.)
10: Masks by setting all SGB color palette color codes to black.
11: Masks by setting all SGB color palette color codes to the same color (color of bit 00).

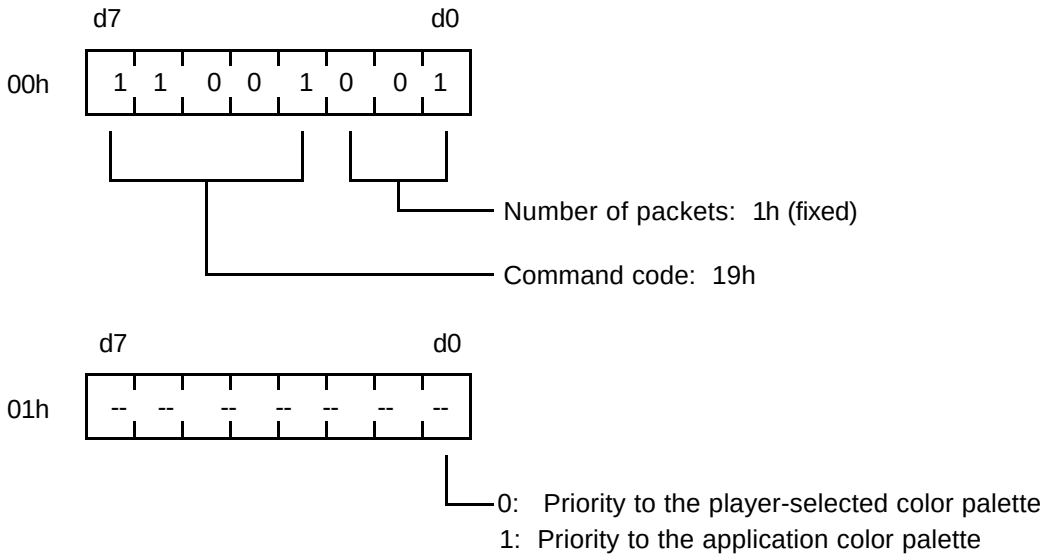
* 2h- Fh filled with 00h.

Note:	***When masking is performed at the start of the game, it should be performed after the DMG reset is canceled and around the time that the DMG screen is displayed on SGB. (The timing of the command should be adjusted so that it is issued after a wait of several frames.) Without this timing, the screen may be momentarily be displayed in monochrome.***

Note:	***Masking with an argument (01h) of 10h or 11h is prohibited during a game.***

Command: PAL_PRI (Code: 19h)

Function: Specifies the priority of the color palette for the application and the color palette selected by the player.



* Default is 0.

Priority to Player-Selected Color Palette

When a screen that uses a player-selected color palette is displayed, any color or attribute settings commands that were sent have no effect on the DMG window.

Priority to Application Color Palette

When a screen that uses a player-selected color palette is displayed and a color or attribute setting command was sent, the sent colors are displayed in the DMG window. The corresponding commands are as follows.

00, 01, 02, 03, 04, 05, 06, 07, 0A, 16 (Code value)

6.3.3 Cautions Regarding Sending Commands

- After each packet (128 bits) is sent, 0 must always be sent in bit 129.
- If a data sequence covers more than 1 packet, byte 1 of each packet after the first is a continuation of the data of the previous packet.
- 00h is written to the unused areas in each packet.
- Note that there are two modes of data transfer: register-file mode and a mode in which 4 Kbytes are tranferred using SGB RAM.
- Controller key input should not be read while a command is being sent.

6.3.4 Sound Flag Summary

- Pre-loaded sound effects A and B can be played simultaneously using system commands.
- The A sound effects are formants, primarily action sounds, and the B sound effects are looping sounds, primarily ambient sounds.
- The interval (frequency) for these sound effects can be set to 4 levels. Changing interval A allows a completely different sound effect to be obtained with the same sound source. In addition, the volume can be set to 3 levels.

6.3.4.1 Sound Effect A Flags

SOUND Command			
01h d7-d0 [bit]		03h d1-d0 [bit]	
Code	Flag Meaning	Recommended Interval Value	Voices Used
00h	Dummy flag for retriggering		• • 6 • 7
80h	Sound effect A stop (mute)		• • 6 • 7
01h	Nintendo	d1 = 1 • d0 = 1	• • • 7
02h	Game over sound	d1 = 1 • d0 = 1	• • 6 • 7
03h	Falling sound	d1 = 1 • d0 = 1	• • • 7
04h	Predetermined sound • • • A	d1 = 1 • d0 = 1	• • 6 • 7
05h	Predetermined sound • • • B	d1 = 1 • d0 = 1	• • 6 • 7
06h	Selected sound • • • A	d1 = 1 • d0 = 1	• • 6 • 7
07h	Selected sound • • • B	d1 = 1 • d0 = 1	• • • 7
08h	Selected sound • • • C	d1 = 1 • d0 = 0	• • 6 • 7
09h	Error sound • • • buzzer	d1 = 1 • d0 = 0	• • • 7
0Ah	Item-catch sound	d1 = 1 • d0 = 0	• • 6 • 7
0Bh	One knock on door	d1 = 1 • d0 = 0	• • 6 • 7
0Ch	Explosion • • • small	d1 = 0 • d0 = 1	• • 6 • 7
0Dh	Explosion • • • medium	d1 = 0 • d0 = 1	• • 6 • 7
0Eh	Explosion • • • large	d1 = 0 • d0 = 1	• • 6 • 7
0Fh	Defeat sound • • • A	d1 = 1 • d0 = 1	• • • 7
10h	Defeat sound • • • B	d1 = 1 • d0 = 1	• • 6 • 7
11h	Striking sound (attack) • • • A	d1 = 0 • d0 = 0	• • 6 • 7
12h	Striking sound (attack) • • • B	d1 = 0 • d0 = 0	• • 6 • 7
13h	Air-sucking sound	d1 = 0 • d0 = 0	• • 6 • 7
14h	Rocket launcher • • • A	d1 = 0 • d0 = 1	• • 6 • 7
15h	Rocket launcher • • • B	d1 = 0 • d0 = 1	• • 6 • 7
16h	Bubbling sound (in water)	d1 = 1 • d0 = 0	• • • 7

SOUND Command			
01h d7-d0 [bit]		03h d1-d0 [bit]	
Code	Flag Meaning	Recommended Interval Value	Voices Used
17h	Jump	d1 = 1 • d0 = 1	• • • 7
18h	Fast jump	d1 = 1 • d0 = 1	• • • 7
19h	Jet (rocket) firing	d1 = 0 • d0 = 0	• • • 7
1Ah	Jet (rocket) landing	d1 = 0 • d0 = 0	• • • 7
1Bh	Cup breaking	d1 = 1 • d0 = 0	• • 6 • 7
1Ch	Glass breaking	d1 = 0 • d0 = 1	• • 6 • 7
1Dh	Level up	d1 = 1 • d0 = 0	• • 6 • 7
1Eh	Air injection	d1 = 0 • d0 = 1	• • • 7
1Fh	Sword wielding	d1 = 0 • d0 = 1	• • • 7
20h	Falling in water	d1 = 1 • d0 = 0	• • • 7
21h	Fire	d1 = 0 • d0 = 1	• • • 7
22h	Breaking wall	d1 = 0 • d0 = 1	• • 6 • 7
23h	Cancellation sound	d1 = 1 • d0 = 0	• • 6 • 7
24h	Stepping	d1 = 0 • d0 = 1	• • 6 • 7
25h	Block-hitting sound	d1 = 0 • d0 = 1	• • 6 • 7
26h	Sound of picture floating into view	d1 = 1 • d0 = 1	• • 6 • 7
27h	Screen fade-in	d1 = 0 • d0 = 0	• • 6 • 7
28h	Screen fade-out	d1 = 0 • d0 = 0	• • 6 • 7
29h	Window opening	d1 = 0 • d0 = 1	• • 6 • 7
2Ah	Window closing	d1 = 0 • d0 = 0	• • 6 • 7
2Bh	Large laser sound	d1 = 1 • d0 = 1	• • 6 • 7
2Ch	Sound of stone door closing (opening)	d1 = 0 • d0 = 0	• • 6 • 7
2Dh	Teleportation	d1 = 1 • d0 = 1	• • • 7
2Eh	Thunder	d1 = 0 • d0 = 0	• • 6 • 7
2Fh	Earthquake	d1 = 0 • d0 = 0	• • 6 • 7
30h	Small laser sound	d1 = 1 • d0 = 0	• • 6 • 7

6.3.4.2 Sound Effect B Flags

SOUND Command			
02h d7-d0 [bit]		03h d5-d4 [bit]	
Code	Flag Meaning	Recommended Interval Value	Voices Used
00h	Dummy flag for retriggering		0 • 1 • 4 • 5
80h	Sound Effects B stop (mute)		0 • 1 • 4 • 5
01h	Applause • • • small crowd	d5 = 1 • d4 = 0	• • • 5
02h	Applause • • • medium crowd	d5 = 1 • d4 = 0	• • 4 • 5
03h	Applause • • • large crowd	d5 = 1 • d4 = 0	0 • 1 • 4 • 5
04h	Wind	d5 = 0 • d4 = 1	• • 4 • 5
05h	Rain	d5 = 0 • d4 = 1	• • • 5
06h	Storm	d5 = 0 • d4 = 1	• 1 • 4 • 5
07h	Hurricane	d5 = 1 • d4 = 0	0 • 1 • 4 • 5
08h	Thunder	d5 = 0 • d4 = 0	• • 4 • 5
09h	Earthquake	d5 = 0 • d4 = 0	• • 4 • 5
0Ah	Lava flow	d5 = 0 • d4 = 0	• • 4 • 5
0Bh	Wave	d5 = 0 • d4 = 0	• • • 5
0Ch	River	d5 = 1 • d4 = 1	• • 4 • 5
0Dh	Waterfall	d5 = 1 • d4 = 0	• • 4 • 5
0Eh	Small character running	d5 = 1 • d4 = 1	• • • 5
0Fh	Horse galloping	d5 = 1 • d4 = 1	• • • 5
10h	Warning sound	d5 = 0 • d4 = 1	• • • 5
11h	Futuristic car running	d5 = 0 • d4 = 0	• • • 5
12h	Jet flying	d5 = 0 • d4 = 1	• • • 5
13h	UFO flying	d5 = 1 • d4 = 0	• • • 5
14h	Electromagnetic waves	d5 = 0 • d4 = 0	• • • 5
15h	Sound of score being raised	d5 = 1 • d4 = 1	• • • 5

SOUND Command			
02h d7-d0 [bit]		03h d5-d4 [bit]	
Code	Flag Meaning	Recommended Interval Value	Voices Used
16h	Fire	d5 = 1 • d4 = 0	• • • 5
17h	Camera shutter (formant)	d5 = 0 • d4 = 0	0 • 1 • 4 • 5
18h	Writing (formant)	d5 = 0 • d4 = 0	• • • 5
19h	Erasing (formant)	d5 = 0 • d4 = 0	• • • 5
81h	Use prohibited (used by system)		
82h	Use prohibited (used by system)		

6.3.4.3 Attributes of A and B Sound Effects

	SOUND Command								
	03h								
	d7	d6	d5	d4	d3	d2	d1	d0	
A Sound Effects					x	x	0	0	Interval (short)
					x	x	0	1	Interval (med-short)
					x	x	1	0	Interval (med-long)
					x	x	1	1	Interval (long)
					0	0	X	x	Volume (high)
					0	1	x	x	Volume (med)
					1	0	x	x	Volume (low)
B Sound Effects	x	x	0	0					Interval (short)
	x	x	0	1					Interval (med-short)
	x	x	1	0					Interval (med-long)
	x	x	1	1					Interval (long)
	0	0	x	x					Volume (high)
	0	1	x	x					Volume (med)
	1	0	x	x					Volume (low)
					1	1			Mute ON

- Mute takes effect only when both bits d2 and d3 are set to 1. If the volume is set for either the A or B sound effect, mute is turned off.
- Fade-out and fade-in take effect with mute-on and mute-off, respectively. Mute-on and mute-off are implemented for BGM played by A and B sound effects and by the APU.
- There is no independent mute-off flag.
- When the mute flag is set, the volume and interval data for the A (Port 1) and B (Port 2) sound effects also should be set.

6.4 Miscellaneous

6.4.1 Reading Input from Multiple Controllers

After a multiplayer request (Command MLT_REQ) is sent, data from controllers 1, 2, 3, and 4 automatically become readable.

In 2-player mode, data from controller 1 is read first, followed by data from controller 2, then data from controller 1 again, and so on. In 4-player mode, the order is controller 1, controller 2, controller 3, controller 4, controller 1 again, and so on.

In these cases, the next controller for which data is to be read must be determined beforehand by reading P10-P13 with P14 and P15 high.

P10 - P13	Next Controller to Read
Fh	Controller 1
Eh	Controller 2
Dh	Controller 3
Ch	Controller 4

Note: ***Controller data cannot be read if Multiplayer 5 and Super NES Mouse are connected at the same time.***

6.4.2 Recognizing SGB

6.4.2.1 Distinguishing Between Game Boy Types (DMG, MGB/MGL, SGB, and SGB2)

The program uses the following methods to determine which of the 4 types is operating.

- Checks the initial value of the internal accumulator of the CPU. (distinguishes between previous/new versions of CPU).

  01 → DMG or SGB
  FF → MGB/MGL or SGB2

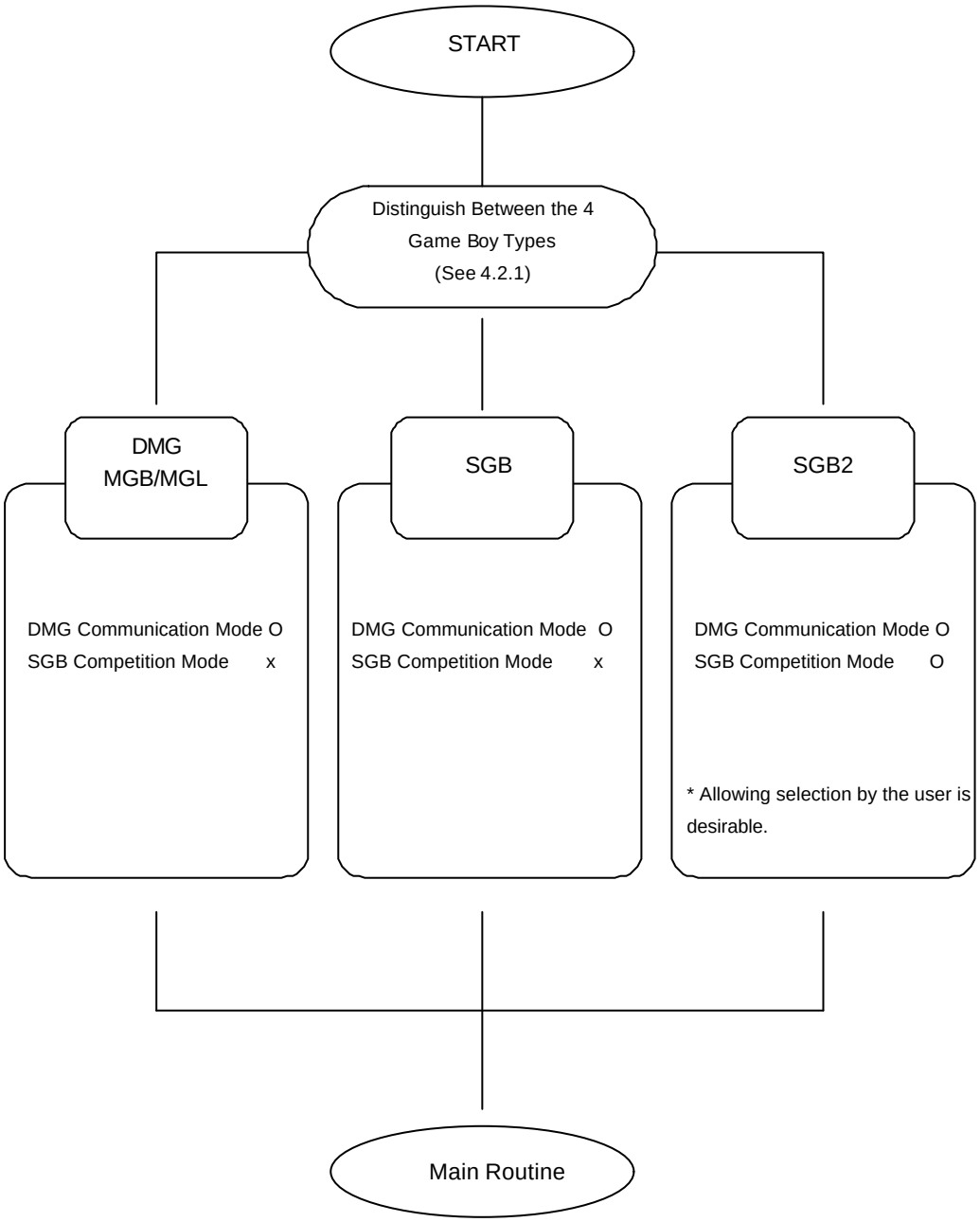
- Sends a muliplayer request (Command MLT_REQ) and determines whether there is a switch to multiplayer mode.

  No switch → DMG or MGB/MGL
  Switch → SGB or SGB2

* The following table summarizes these methods.

Initial Value of CPU Internal Accumulator	Switch/No Switch to Multiplayer Mode	Game Boy Type
01	No switch	DMG
	Switch	SGB
FF	No switch	MGB/MGL
	Switch	SGB2

6.4.2.2 Usage Example: Distinguishing Between the 4 Game Boy Types



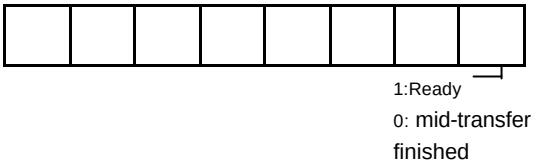
* A sample program for distinguishing between Game Boy types is provided.

6.4.3 SGB Register Summary

The following registers can be used to perform functions such as resetting the SGB CPU from a program transferred to Super NES W-RAM and receiving and passing data to a DMG program.

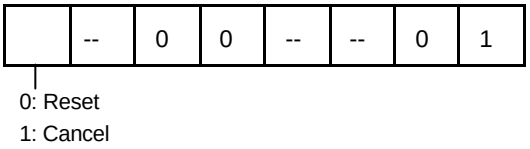
Register File Status
[RFS] 6002h (RD)

Reads the status of the register file



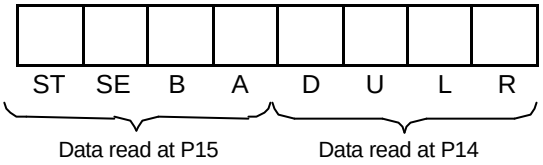
DMG Reset Register
[DRR] 6003h (WR)

Resets the SGB CPU



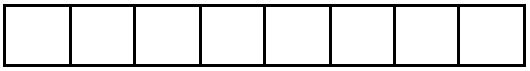
Controller Register1
[CR1] 6004h (WR)

Writes data from controller 1



Register File
[RF_0 - RF_F] 7000h - 700Fh (RD)

Register file for communication



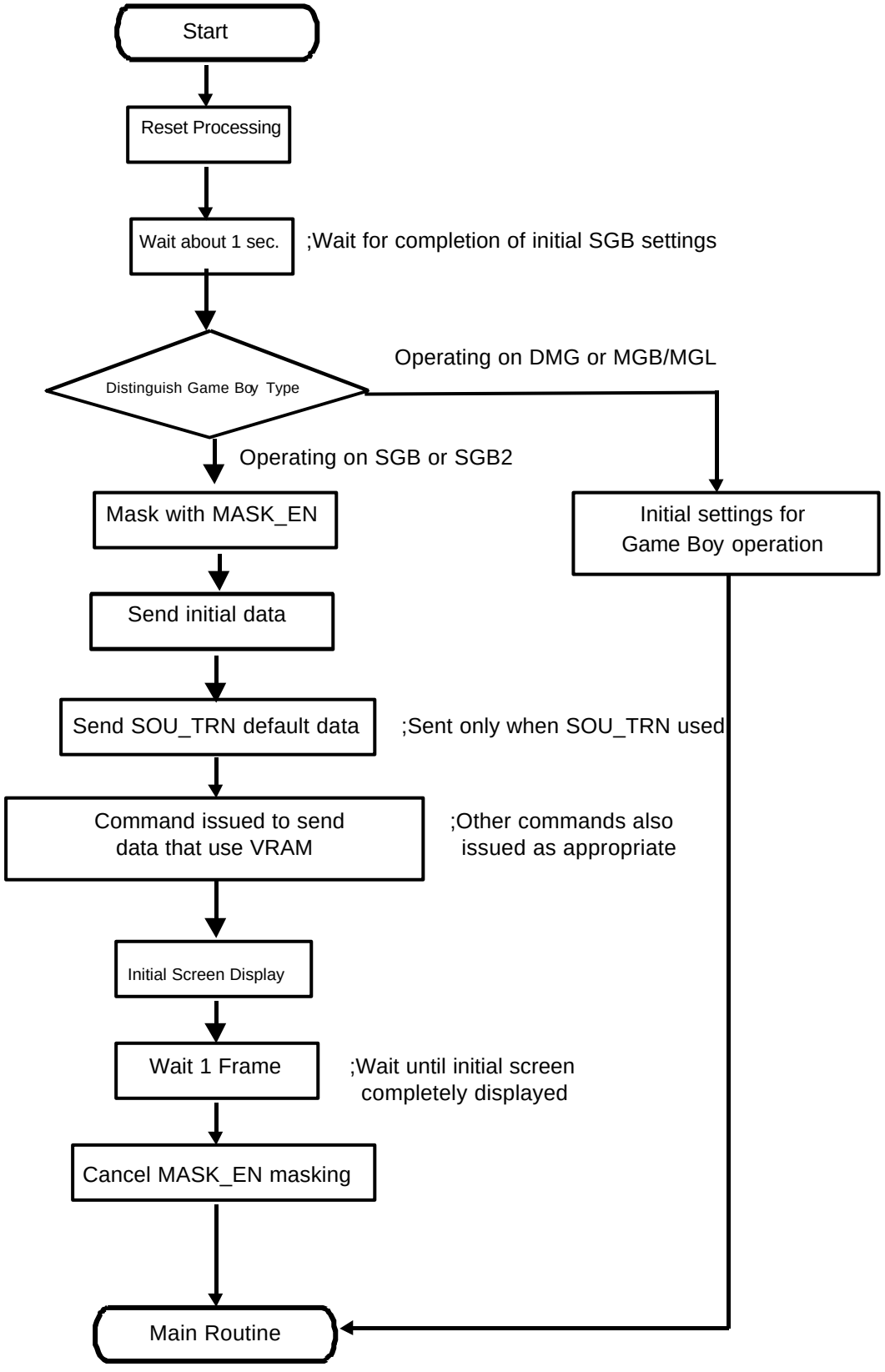
* The RFS flag (6002h) is cleared to 0 by the reading of 7000h.

The SGB CPU can be reset using DRR.

Using RF_0 - RF_F and RFS allows data sent to the register file by the DMG program to be received by the Super NES program.

CR1 is a register used by the original SGB system program for writing keypad data from controller 1. The Super NES program can use the controller-reading routine of the DMG program to receive data written to this register.

6.4.4 Flowchart of Initial Settings Routine


Start
Reset Processing
Wait about 1 sec.
;Wait for completion of initial SGB settings
Distinguish Game Boy Type
Operating on DMG or MGB/MGL
Operating on SGB or SGB2
Mask with MASK_EN
Initial settings for Game Boy operation
Send initial data
Send SOU_TRN default data
;Sent only when SOU_TRN used
Command issued to send data that use VRAM
;Other commands also issued as appropriate
Initial Screen Display
Wait 1 Frame
;Wait until initial screen completely displayed
Cancel MASK_EN masking
Main Routine

6.5 Programming Cautions

6.5.1 ROM Registration Data

To use SGB functions (system commands), the following values must be stored at the ROM addresses indicated.

146h ← 03h and 14Bh ← 33h

6.5.2 Initial Data

When writing programs that use the system commands of SGB and SGB2, use the initialization routine of the game program to send the following 8 packets of default data to the register file.

```
INIT1   DEFB    $79,$5D,$08,$00,$0B,$8C,$D0,$F4,$60,$00,$00,$00,$00,$00,$00,$00
INIT2   DEFB    $79,$52,$08,$00,$0B,$A9,$E7,$9F,$01,$C0,$7E,$E8,$E8,$E8,$E8,$E0
INIT3   DEFB    $79,$47,$08,$00,$0B,$C4,$D0,$16,$A5,$CB,$C9,$05,$D0,$10,$A2,$28
INIT4   DEFB    $79,$3C,$08,$00,$0B,$F0,$12,$A5,$C9,$C9,$C8,$D0,$1C,$A5,$CA,$C9
INIT5   DEFB    $79,$31,$08,$00,$0B,$0C,$A5,$CA,$C9,$7E,$D0,$06,$A5,$CB,$C9,$7E
INIT6   DEFB    $79,$26,$08,$00,$0B,$39,$CD,$48,$0C,$D0,$34,$A5,$C9,$C9,$80,$D0
INIT7   DEFB    $79,$1B,$08,$00,$0B,$EA,$EA,$EA,$EA,$EA,$A9,$01,$CD,$4F,$0C,$D0
INIT8   DEFB    $79,$10,$08,$00,$0B,$4C,$20,$08,$EA,$EA,$EA,$EA,$EA,$60,$EA,$EA
```

6.5.3 SOU_TRN initial Data

When using the SOU_TRN system command, send the following 5 packets of SOU_TRN default data to the register file before SOU_TRN is used.

```
STI
DB      $79, $00, $09, $00, $0B
DB      $AD, $C2, $02, $C9, $09, $D0, $1A, $A9, $01, $8D, $00

ST2
DB      $79, $0B, $09, $00, $0B
DB      $42, $AF, $DB, $FF, $00, $F0, $05, $20, $73, $C5, $80

ST3
DB      $79, $16, $09, $00, $0B
DB      $03, $20, $76, $C5, $A9, $31, $8D, $00, $42, $68, $68

ST4
DB      $79, $21, $09, $00, $01
DB      $60, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00

ST5
DB      $79, $00, $08, $00, $03
DB      $4C, $00, $09, $00, $00, $00, $00, $00, $00, $00, $00
```

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 7 SUPER GAME BOY SOUND 203

7.1 SGB Sound Program Overview..... 203

7.2 Memory Mapping (SUPER NES APU)..... 204

7.3 Creating and Transferring Score Data..... 205

7.3.1 Transferring Score Data.....205
7.3.2 Summary of BGM Flags 205
7.3.3 Overview of Creating Score Data 206
7.3.4 Setting the NEWS System Working Environment.....206
7.3.5 Setting the Working Environment Using IS-SOUND.....209
7.3.6 Score Data Format When Using Original Tools..... 211
7.3.7 Cautions Regarding Production of Musical Pieces 222
7.3.8 Format of Transferred Data 223

7.4 SGB Sound Program Source List..... 225

7.5 Transferring Audio Data to the Score Area..... 228

7.5.1 Required Data and Procedure for Audio Output.....228
7.5.2 Transfer File Example.....229

THIS PAGE WAS INTENTIONALLY LEFT BLANK

CHAPTER 7: SUPER GAME BOY SOUND

7.1 SGB Sound Program Overview

The SGB sound program is a special SGB program built into the SGB system program. The sound program is automatically transferred to the Super NES APU at system startup.

Using the SGB system commands, pre-loaded sound effects in the sound program can be used in Game Boy application programs that support SGB (SGB software).

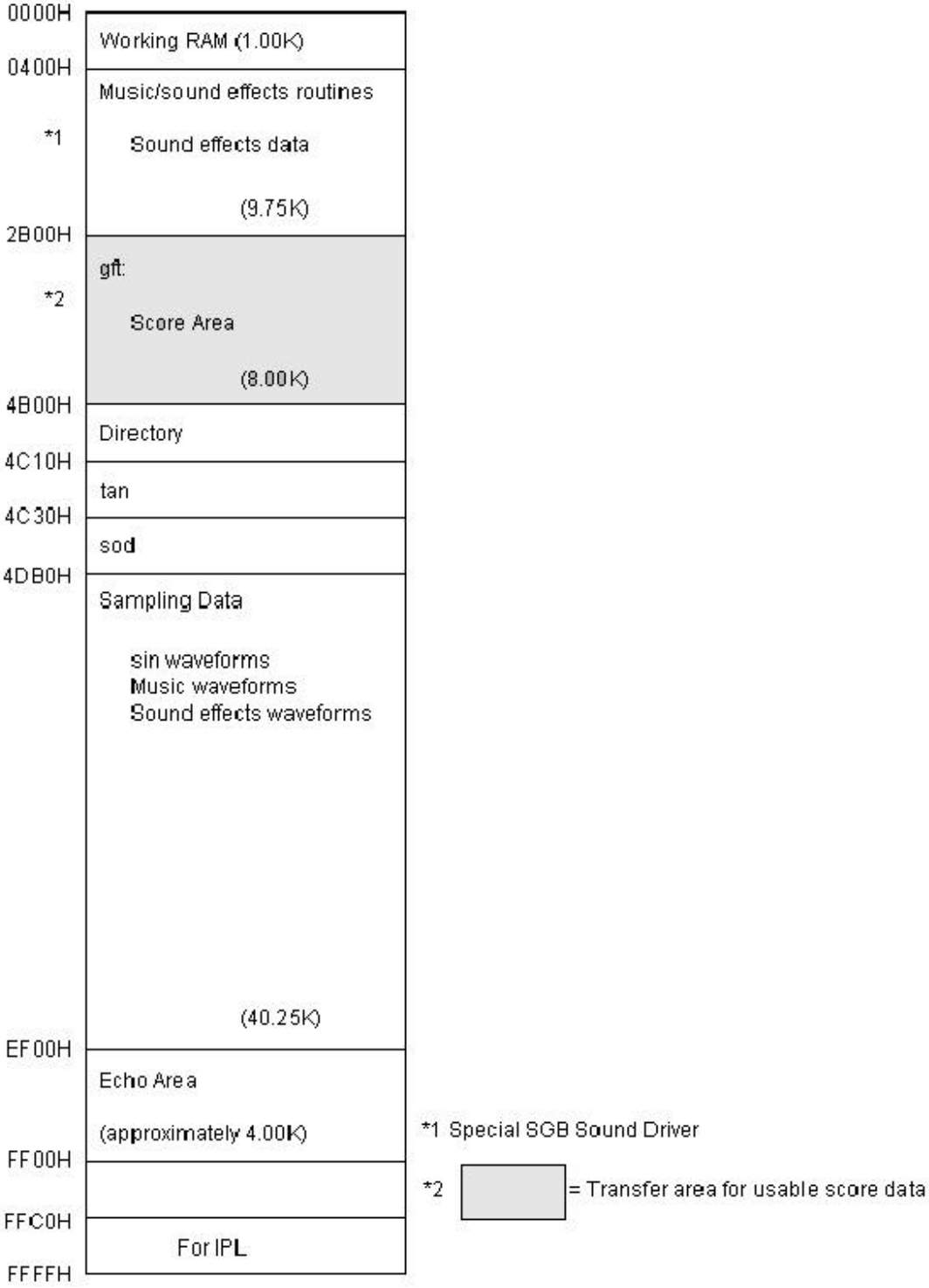
These commands can be used to set each of the 73 types of pre-loaded sound effects to 4 intervals (playback frequencies) and 3 volume levels.

Also preloaded are music data for BGM (instruments sound sampling data). This easily allows play of score data created with Kankichi-kun, the tool for creating Super NES scores, and score data for KAN.ASM, the standard driver that is a software tool for IS-SOUND.

In addition, information on the SGB score data format has been made openly available, allowing those using tools other than the NEWS system or IS-SOUND to create score data in this format.

7.2 Memory Mapping (Super NES APU)

[APU Addresses]



7.3 Creating and Transferring Score Data

7.3.1 Transferring Score Data

BGM can be played with the APU by using the SOU_TRN command to transfer original score data to the prescribed area of APU RAM. The user area is the 8 Kbytes from 2B00h to 4AFFh.

7.3.2 Summary of BGM Flags

SOUND Command	
04h d7-d0 [bit]	
Code	Flag Description
00h	Dummy flag for retriggering
01h | 0Fh	
80h	BGM stop flag
FEh	Use prohibited (used by system)
FFh	Use prohibited (used by system)

Note ***If 01h-0Fh are set without score data being transferred, the BGM built into the system is played.***

This BGM is exclusively for use by the system, so 01h-0Fh should not be written as a BGM flag without original score data being transferred.

Even if original score data is transferred, there is risk that the sound program will run uncontrolled if a non-designated code is written.

Muting is in effect when the system is initialized, so the BGM playback settings must be made after muting is canceled.

7.3.3 Overview of Creating Score Data

Original BGM can be played with the SGB sound program by transferring score data to the APU using system commands.

Fifty-seven sounds can be used in BGM, and the score data can be up to just under 8 Kbytes in size.

The method used to create a musical piece is nearly identical to that of the standard Super NES.

When the NEWS system is used, score data is created using Kankichi-kun. When IS-SOUND is used, score data created by an external sequencer are processed through MIDI and converted to create score data supported by the standard sound driver KAN.ASM.

In addition the SGB score data format has been made openly available, allowing those using original tools to create score data in this format.

In creating musical pieces, please refer to Section 7.4, *SGB Sound Program Source List*, when selecting sounds. Please do not change the order of these source data.

7.3.4 Setting the NEWS System Working Environment

Working Environment Settings for the NEWS System

1. Rename the current sobox directory.

   % mv sobox xxxxx

2. Create a new sobox directory.

   % mkdir sobox

   * SGB can use only specific sound objects. Thus, special SGB source data must be installed. A sobox directory for SGB use must be created to prevent loss of previously installed source data files with the same names as the data files to be installed.

3. Move to the sobox directory.

4. From the installation disk, install soread in this directory.

   % tar xvf /dev/rfh0a soread

5. Next install the sampling data files (xxx.so . . .) in this directory.

   % soread

   Executing the above command causes the sampling data to be automatically installed.

6. Create a new SGB working directory at any location.

   % mkdir #####

7. Move to the SGB working directory.

8. From the installation disk, install the following files in the working directory: **sgbt.asm**, **sample.kan**, **check.kan**, **kankichib.hex**, and **kan.equ**.

   % tar xvf /dev/rfh0a sgbt.asm sample.kan check.kan kankichib.hex kan.equ

   * The organization and address settings in **kankichib.hex** are as shown below.

   * Use the installed **kankichib.hex** file when starting up **mapu**.

Item	Setting
Kan.equ	4c30h
Kan.tan	04c10h
Program start address	00400h
DIR address	04b00h
Echo end address	0ff00h
Sound score start address	02b00h

9. Make the following changes in the file .cshrc in the home directory.

--- Following are the Sound Generation Environments Settings ---

	Before Change	After Change
StartOfKan	0x800h	0x400h
StartOfDirectory	0x3c00h	0x4b00h
EndOfDirectory	0x3cffh	0x4c0fh
StartOfAttribute	0x3e00h	0x4c30h
StartOfTan	0x3f00h	0x4c10h
StartOfWave	0x4000h	0x4db0h
EndOfWave	0xcfffh	0xeeffh
StartOfFumen	0xd000h	0x2b00h

10. In the home directory, execute the following command: **source .cshrc**.

Cautions When Using Kankichi-kun

1. Copy **sample.kan** to a newly created score data file, **[score_name].kan**.

   ```
   % cp sample.kan xxx.kan
   ```

   * This avoids the task of creating a source list in source-list order when using mapu.

2. Start **mapu**.

   ```
   % mapu -k
   ```

   * When starting mapu for the first time, press the NICE reset button.

3. The usable sounds (sources) can be checked with **mapu**. Selecting **check.kan** allows the sounds to be checked in source-list order.

   * If data in files such as **check.kan** are changed, the sounds cannot be checked.

4. To actually create a tune, select **xxx.kan**.

   * Source data (sampling data) that SGB can use have been set in **xxx.kan**. The source list is shown in Section 7.4, *SGB Sound Program Source List*. Note that changing the order of the source list will result in sounds different from the intended sounds when BGM is played.

5. When producing a musical piece, see Section 7.3.7, *Cautions Regarding Production of Musical Pieces*. Refer to the Kankichi-kun Manual.

6. Finally, convert to the file format described in Section 7.3.8, *Format for Transferred Files*.

7.3.5 Setting the Working Environment Using IS-SOUND

Environment Required

- Hardware: IS-SOUND connected to a host computer
- Software: IS-SOUND software tools (installed)

Revisions

1. Portions of the IS-SOUND software tool KAN.EQU were revised as indicated below (older versions only).

Before Revisions	After Revisions
cut: equ 122+ 0x80h fft: equ 123+ 0x80h ply: equ 124+ 0x80h wav: equ 125+ 0x80h sel: equ 126+ 0x80h	wav: equ 122+ 0x80h sel: equ 123+ 0x80h cut: equ 124+ 0x80h fft: equ 125+ 0x80h ply: equ 126+ 0x80h

2. Set Gate Table data to 050 · 101 · 127 · 152 · 178 · 203 · 229 · 252.
3. Set Velocity Table data to 025 · 050 · 076 · 101 · 114 · 127 · 140 · 152 · 165 · 178 · 191 · 203 · 216· 229 · 242 · 252.

Note ***Sound data (sampling data) are required to check music data using IS-SOUND. Consequently, a program equivalent to the sound program built into the SGB hardware (including sound-effect data) and sampling data (sound data) have been provided in a hex file for MS-DOS. The following briefly describes how to set up this program and data.***

Setting the Working Environment

1. Create an SGB working directory at any location, and move to that directory.
2. Copy **sgbsound.hex** from the disk to the working directory.
3. Start the debugger **shvc**.
4. Also start the sound debugger **ssnd**.
5. Execute r sgbsound.hex to load **sgbsound.hex**.
6. Execute **g400** to run the sound program.
7. Press the HOME button to switch to shvc mode.
8. Execute **s2140** to write 01 (from the main program, writes 01 to 0 of the sound port).

With this procedure, the pre-loaded source data (sampling data) are played in the order shown in Section 7.4 of this chapter, *SGB Sound Program Source List*.

After the data is transferred once, only the score data needs to be transferred to allow music to be checked again.

Cautions

1. Score data is the data defined in KAN.ASM Version 1.21 as being located from GFT onward. For information on all items related to converting data from other sequencers to score data, formats, and tool usage, see the IS-SOUND manual.

2. Set the source data number according to the source list.

3. Set the starting address of the score data to 2B00h.

4. When producing a musical piece, do so in accordance with Section 7.3.7, *Cautions Regarding Production of Musical Pieces*.

5. Convert to the file format described in Section 7.3.8, *Format for Transferred Files*.

7.3.6 Score Data Format When Using Original Tools

The score data format has been made openly available for the benefit of those using original development tools.

Data that is not in this format will not operate on SGB.

Note that in some cases, program control may be lost.

Score Data

Glossary of Terms

Gft	Location of tune table definitions (collection of tune label definitions). Up to 15 tunes can be defined. The order defined here corresponds to the flag set for port 0 (0x01-0x0F).
Tune label	A label name applied to each tune.
Block	A unit several bars long that each tune is divided into.
Parts	The channels that make up each block (maximum of 8 parts).
Performance data	The aggregate of the score data played by the parts. The parts in the channels must all be the same length (number of steps) in a given block.

Overall Format of Score Data

Example 1: (Area inside frame = Data table for 1 tune)

```
                org  02b00H                          ; (a) Starting address of score data
gft:                                                 ; (b) Tune table
                dw  bgm1,bgm2, ···                   ; Indicate the tune labels

              bgm1:       ; (c) Tune label 1
                dw  bgm1_block1                              ; (d) Block 01
bgm1_0:
                dw  bgm1_block2                              ; (d) Block 02
                dw  bgm1_block3                              ; (d) Block 03
                dw 255                               ; (e) Repetition code (endless)
                dw  bgm1_0                                   ; (e) Repetition starting address
                dw  000                              ; (f) Tune label end code
              ;
 bgm1_block1:                                        ; (g) Block 01
                dw  bgm1_block1_0                    ; (g) Starting address of Part 0
                dw  bgm1_block1_1                    ; (g) Starting address of Part 1
                dw  bgm1_block1_2                    ; (g) Starting address of Part 2
                dw  bgm1_block1_3                    ; (g) Starting address of Part 3
                dw  00                               ; (g) Part 4 unused
                dw  00                               ; (g) Part 5 unused
                dw  00                               ; (g) Part 6 unused
                dw  00                               ; (g) Part 7 unused
bgm1_block2:
                 .    .                                      ; (g) Same in Block 2
                 .    .
bgm1_block3:
                 .    .                                      ; (g) Same in Block 3
                 .    .
bgm1_block1_0:                    ; Block 01        (h) Part 0 performance data
                db  tp1,049,mv1,200,sno,$1a,pv1,180,pan,010
                db  ecv,255,040,040,edl,002,090,002,tun,050
                db  012,P99+V99,c30,d30,e30,f30,024,g30,kyu
                 db  00                                      ; (h)  Part end code
 bgm1_block1_1:                                              ; (h)  Part 1 performance data
                db  sno,$1b,pv1,140,pan,008,tun,030
                db  096,P99+V99,g20
bgm1_block1_2:                                              ; (h) Part 2 performance data
                db  sno,$1b,pv1,140,pan,008,tun,030
                db  096,P99+V99,e20
bgm1_block1_3:                                              ; (h) Part 3 performance data
               db  sno,$1b,pv1,140,pan,008,tun,030
               db  096,P99+V99,c20
bgm1_block2_0:                          Block 02        (h) Part 0 performance data
                 db  sno,$1a,pv1,200,pan,012,tun,050
                 db • • • • • • • • • • • • • • • • • • •
                 db  00
                  •
bgm1_block3_0:       Block03    (h) Part 0 performance data
              db sno,$1a,pv1,200,pan,012,tun,050
              db • • • • • • • • • • • • • • • • • • •
                 db  00
                   •
```

Description of Example 1

(a) The score data map to memory addresses 2B00h-4AFFh in the APU.
If this area is exceeded, a portion of the sound program will be destroyed.

(b) gft: is the starting address of the entire tune table.
dw, bgm1, and bgm2... are the tune labels and the starting addresses of the score data items.

(c) The tune label.
The order in which the blocks are played is defined following the tune label.
The dotted frame encloses the data for one tune, bgm1.

(d) Data for each block.

(e) 01h-7Fh (01-127) is the number of loops (repetitions); 82h-FFh (130-255) is an endless loop.
If repetition is not needed, set the end code (00h) instead of a loop code.

(f) Block definition end code.

(g) Location where the parts of each block are indicated and the part labels are defined.

Defines the part labels for parts 0, 1, 2, . . . 7 in ascending order from top to bottom. 00h should be written for unused parts. Even if some parts are unused, always define 8 parts.

(h) The performance data for each part.

Play Data Overview

Parameters such as temp, volume, pan, source number, echo, velocity, interval, and sound length are set here.

For specific descriptions, see *Code Summaries* on the next page.

First set are the effects parameters – such as main volume, ramp, and echo – for Part 0 of the first block. Once these are set, they need not be set again (for other blocks or parts) as long as they are not changed.

Next the parameters such as part volume, pan, source number, and tuning are set for each part.

Then the sound length, velocity + gate time, and interval are set in that order. Be careful to ensure that sound length is always set first, followed by velocity + gate item, then the interval.

If the next sound is the same as the previous sound, the sound length, velocity, and gate time need not be set again.

Finally, a data end code of 00 is set for Part 0 of each block.

Settings for parts 1-7 are not required.

The lower parts and blocks are set in the same manner.

Code Summaries

a) Length Data (step time)
This is the length (step time) to the subsequent sound; it corresponds to the length of the sound envelope. The code corresponding to each sound envelope is shown in the following table. Please use the appropriate code in the settings.

Note Length	Code	Note Length	Code	Note Length	Code
Sixteenth note	6	Dotted eighth note	18	Half note	48
Dotted sixteenth note	9	Quarter note	24	Dotted half note	72
Eighth note	12	Dotted quarter note	36	Whole note	96

Note: For triplets and thirty-second notes, convert using the above values.

b) Velocity (volume) + gate time
Velocity expresses the volume as a percentage. Here it can be set to 16 levels using the lower-order 4 bits (d0 – d3).
Gate time expresses as a percentage the length that the sound is actually emitted. It can be set to 8 levels using the higher-order 3 bits (d4 – d6).
Changing these values provides legato and staccato effects.
The following table lists the values defined by the SGB sound driver.
The settings are designated using the codes for the listed velocities (VELOCITY) and gate times (GATE_TIME).

Symbol	Code	d7	d6	d5	d4	d3	d2	d1	d0	Rate
V10	0x00h	x	x	x	x	0	0	0	0	VELOCITY=010%
V20	0x01h	x	x	x	x	0	0	0	1	VELOCITY=020%
V30	0x02h	x	x	x	x	0	0	1	0	VELOCITY=030%
V40	0x03h	x	x	x	x	0	0	1	1	VELOCITY=040%
V45	0x04h	x	x	x	x	0	1	0	0	VELOCITY=045%
V50	0x05h	x	x	x	x	0	1	0	1	VELOCITY=050%
V55	0x06h	x	x	x	x	0	1	1	0	VELOCITY=055%
V60	0x07h	x	x	x	x	0	1	1	1	VELOCITY=060%
V65	0x08h	x	x	x	x	1	0	0	0	VELOCITY=065%
V70	0x09h	x	x	x	x	1	0	0	1	VELOCITY=070%
V75	0x0Ah	x	x	x	x	1	0	1	0	VELOCITY=075%
V80	0x0Bh	x	x	x	x	1	0	1	1	VELOCITY=080%
V85	0x0Ch	x	x	x	x	1	1	0	0	VELOCITY=085%
V90	0x0Dh	x	x	x	x	1	1	0	1	VELOCITY=090%
V95	0x0Eh	x	x	x	x	1	1	1	0	VELOCITY=095%
V99	0x0Fh	x	x	x	x	1	1	1	1	VELOCITY=099%
P20	0x00h	x	0	0	0	x	x	x	x	GATE_TIME=020%
P40	0x10h	x	0	0	1	x	x	x	x	GATE_TIME=040%
P50	0x20h	x	0	1	0	x	x	x	x	GATE_TIME=050%
P60	0x30h	x	0	1	1	x	x	x	x	GATE_TIME=060%
P70	0x40h	x	1	0	0	x	x	x	x	GATE_TIME=070%
P80	0x50h	x	1	0	1	x	x	x	x	GATE_TIME=080%
P90	0x60h	x	1	1	0	x	x	x	x	GATE_TIME=090%
P99	0x70h	x	1	1	1	x	x	x	x	GATE_TIME=099%

Symbol input example: P99+V99 Code input example: 70h+ 0Fh

*When setting score data using symbols, assemble after defining the equal statement according to the table above.

c) Interval Data
Intervals for 6 octaves can be set here.
Depending on the sound, however, high sounds may not be heard.
The following table shows the correspondence between code settings and intervals. Please refer to this table when setting an interval.

*Interval symbols: C01- B50
Codes: 81h-C7h (tie = C8h · rest = C9h)

*When score data is set using symbols, assemble after defining the equals statement.

Octave 0	
Interval	Code
C01	81h
D00	82h
D01	83h
E00	84h
F00	85h
F01	86h
G00	87h
G01	88h
A00	89h
A01	8Ah
B00	8Bh

Octave 1	
Interval	Code
C10	8Ch
C11	8Dh
D10	8Eh
D11	8Fh
E10	90h
F10	91h
F11	92h
G10	93h
G11	94h
A10	95h
A11	96h
B10	97h

Octave 2	
Interval	Code
C20	98h
C21	99h
D20	9Ah
D21	9Bh
E20	9Ch
F20	9Dh
F21	9Eh
G20	9Fh
G21	A0h
A20	A1h
A21	A2h
B20	A3h

Octave 3	
Interval	Code
C30	A4h
C31	A5h
D30	A6h
D31	A7h
E30	A8h
F30	A9h
F31	AAh
G30	ABh
G31	ACh
A30	ADh
A31	AEh
B30	AFh

Octave 4	
Interval	Code
C40	B0h
C41	B1h
D40	B2h
D41	B3h
E40	B4h
F40	B5h
F41	B6h
G40	B7h
G41	B8h
A40	B9h
A41	BAh
B40	BBh

Octave 5	
Interval	Code
C50	BCh
C51	BDh
D50	BEh
D51	BFh
E50	C0h
F50	C1h
F51	C2h
G50	C3h
G51	C4h
A50	C5h
A51	C6h
B50	C7h

Misc.			
Tie	TIE	C8h	See Note 2
Rest	KYU	C9h	See Note 3

Note 1: A value of 1 in the right-most position of the interval symbol indicates a # (sharp).
A *b* (flat) is represented as the # of one interval lower.
Example: C01=C# for interval 0.

Note 2: When specifying a tie, first set the step time (length) and velocity + gate time. (This can be skipped if unchanged from the previous sound.)

A tie cannot be used at the start of a block.

Note 3: When specifying KYU (a rest), first set the step time (length). (This can be skipped if unchanged from the previous sound.)

Settings Example:

```
        Length   Gt & Vel   Interval      Code
db        024,   P99+V99,      C30   ;(0A4h) for specifying an interval
        Length   Gt & Vel      Tie      Code
db        048,   P90+V95,      TIE   ;(0C8h) for specifying a tie
        Length     Rest                Code
db        096      KYU                ;(0C9h) for specifying a rest
```

d) Special Symbols
The special symbols represent special data for implementing a variety of special effects. These include sound change, crescendo, panpot change, vibrato, tremolo, and echo. Each symbol has its own parameters.

The following table lists these special symbols, their parameters, and the valid values for these parameters.

☆ Special Symbols Summary No. 1

Symbol Code	First Argument (range)	Second Argument (range)	Third Argument (range)	Function
sno ($E0)	SOURCE NAME 0 ≤ X ≤ 127			Sound change
pan ($E1)	Pan value 0 ≤ X ≤ 20			Panpot (0=L/20=R/10=C) (10 = default)
pam ($E2)	No. of steps 1 ≤ X ≤ 255	Pan value 0 ≤ Y ≤ 20		Move panpot (Y takes effect after X steps)
vib ($E3)	No. of hold steps 0 ≤ X ≤ 255	Rate 1 ≤ Y ≤ 255	Depth 1 ≤ Z ≤ 255	Vibrato (no. of hold steps is the time till vibrato takes effect)
vof ($E4)				Vibrato off
☆ mv1 ($E5)	Volume 0 ≤ X ≤ 255			Main volume (192 = Default value)
☆ mv2 ($E6)	No. of Steps 1 ≤ X ≤ 255	Volume 0 ≤ Y ≤ 255		Move main volume (used for crescendo/decrescendo) (Y takes effect after X steps)
☆ tp1 ($E7)	Rate 1 ≤ X ≤ 82			Tempo See Note 1.
☆ tp2 ($E8)	No. of steps 1 ≤ X ≤ 255	Rate 1 ≤ Y ≤ 82		Move tempo (Used for retardando/accelerando) (Y takes effect after X steps)
☆ ktp ($E9)	Transposition level $E8≤F≤$FF(–value) $00≤X≤$18			Main key transpose (1= semitone up/-1= semitone down) – is the two's complement
ptp ($EA)	Transposition level $E8≤X≤$FF(–value) $00≤X≤$18			Part key transpose (1= semitone up/-1= semitone down) – is the two's complement

☆ Special Symbols Summary No. 2

Symbol Code	First Argument (range)	Second Argument (range)	Third Argument (range)	Function
tre ($EB)	No. of hold steps 0 ≤ X ≤ 255	Rate 1 ≤ Y ≤ 255	Depth 1 ≤ Z ≤ 255	Tremelo (no. of hold steps is the time till tremelo takes effect)
tof ($EC)				Tremelo off
pv1 ($ED)	Volume 0 ≤ X ≤ 255			Part volume (192=Default value)
pv2 ($EE)	No. of steps 1 ≤ X ≤ 255	Volume 0 ≤ Y ≤ 255		Move part volume (Used for crescendo/decrescendo) (Y takes effect after X steps.)
pat ($EF)	PAT ADRS(L) $00 ≤ X ≤ $FF	PAT ADRS(H) $00 ≤ Y ≤ $FF	REPEAT PAT 1 ≤ Z ≤ 255	Pattern data subroutine Seen Note 2.
vch ($F0)	No. of steps 1 ≤ X ≤ 255			Vibrato deepens gradually over X number of steps
swk ($F1)	No. of hold steps 0 ≤ X ≤ 255	No. of steps 1 ≤ Y ≤ 255	Amount of change $DC ≤ Z ≤ $FF(– value) $00 ≤ Z ≤ $24	Start sweep from next sound – is the two's complement
sws ($F2)	No. of hold steps 0 ≤ X ≤ 255	No. of steps 1 ≤ Y ≤ 255	Amount of change $DC ≤ Z ≤ $FF(– value) $00 ≤ Z ≤ $24	Start sweep heading into next sound – is the two's complement
sof ($F3)				Sweep off
tun ($F4)	Amount of change 0 ≤ X ≤ 255			Tune (Semitone up with 255)
ecv ($F5)	ECHO CHANNEL 0 ≤ X ≤ 255	ECHO-VOL(L) 0 ≤ Y ≤ 255	ECHO-VOL(R) 0 ≤ Z ≤ 255	Echo volume Seen Note 3.
eof ($F6)				Echo off
edl ($F7)	ECHO TIME 1 ≤ X ≤ 15	FEED BACK $9D ≤ Y ≤ $FF (– value) $00 ≤ Y ≤ $7F	FILTER No. 0 ≤ Z ≤ 10	Echo delay See Note 4. – is the two's complement

Symbol Code	First Argument (range)	Second Argument (range)	Third Argument (range)	Function
ev2 ($F8)	No. of steps 1≤X≤255	ECHO-VOL(L) 0≤Y≤255	ECHO-VOL(R) 0 ≤ Z ≤ 255	Move echo volume (YZ values take effect after X steps)
swp ($F9)	No. of hold steps 0 ≤ X ≤ 255	No. of steps 1 ≤ Y ≤ 255	SWEEP value interval	Sweep (once) The interval takes effect after the specified number of hold steps.

Note 1: The tempo values set by the program data and the actual (musical piece) tempos that correspond to those values are as follows.

Please refer to this table to make the conversions.

Music Tempo	Driver Tempo	Music Tempo	Driver Tempo
Quarter note = 400	82	Quarter note = 120	25
Quarter note = 300	62	Quarter note = 60	12
Quarter note = 240	49	Quarter note= 30	6

Note 2: Used when the same performance data is repeated (for data compression). Following the pat code, the L and H addresses and the repetition frequency for the performance data is set. The performance data at the addresses specified by pat are then read. The data is played the number of times specified by the repetition frequency. The performance data at the locations specified by pat require an end code of 00h.

Note 3: When applying echo, **ecv** and **edl** are required. The value entered for the echo channel is 1 for echo used in Part 0, 2 for Part 1, 4 for Part 2, 8 for Part 3, 16 for Part 4, 32 for Part 5, 64 for Part 6, and 128 for Part 7. When echo is used for multiple parts, enter the sum of the channel number values.

Examples:

When echo is used for parts 0 and 1, the value entered is 3.
When echo is used for all parts, the value entered is 255.

Note 4: Echo time is the delay duration. It uses RAM area equal to twice the echo time value, expressed in Kbytes. The echo area in SGB is 4 Kbytes, so a value of 2 or less should be entered. Feedback indicates the amount of delay returned. Filter No. indicates the type of filter applied to the delayed sound.

0 = no filter; 1 = high-pass filter; 2 = low-pass filter; 3 = band-pass filter

*The symbols marked with a ☆ in the *Special Symbols* table are applied to all parts. These should be set in the first part.

*When using a symbol to set a special symbol for score data, assemble after defining the equals statement according to the *Special Symbols* table.

*The special symbols and the arguments that follow should be set in the order shown in the tables.

*If using IS-SOUND, load **sgbsound.hex** according to the steps in Section 7.3.5, *Setting the Working Environment Using IS-SOUND*. Transferring the subsequently created score data allows the tunes and sounds to be checked.

Cautions

1. The starting address for score data should be set to 2B00h.

2. Source numbers should be set according to the source list.

3. Musical pieces should be produced according to the instructions in Section 7.3.7, *Cautions Regarding Production of Musical Pieces*.

4. Convert to the file format described in Section 7.3.8, *Format for Transferred Files*.

Summary of Play Data Codes

00h	Part end code
01h-7Fh	Note/rest length data & **VELOCITY** (volume) + **GATE_TIME**
80h-C7h	Interval (sound length) data (C00-B50) * C01-B50 in SGB
C8h	Tie (**TIE**)
C9h	Rest (**KYU**)
CAh-DFh	Use prohibited
E0h-F9h	Special symbols
FAh-FFh	Use prohibited

7.3.7 Cautions Regarding Production of Musical Pieces

The echo parameters set in BGM are applied in the same manner for the A and B sound effects. This is because echo is applied equally to all 8 channels. The parameters have been tuned so that they can also be used with BGM, so please note this when resetting the parameters.

Score Data Settings			
Special Symbol	Echo Channel	Echo Volume L	Echo Volume R
ecv	OOO (Note 1)	40	40
Special Symbol	Echo Time	Feed Back	Filter No.
edl	2 (Note 2)	90	2

Note 1: If echo is not used, specify **eof** (special symbol) instead of **ecv**.

Note 2: If a value greater than 2 is specified for **Echo Time**, the sampling data will be destroyed. Up to 15 tunes can be registered (01h-0Fh). Channels 2 and 3 are allocated for BGM, so these channels should be used for regular playback of BGM parts.

Microtuning of source data used for notes should be specified using the **tun** code with the score data. For tuning values, refer to the recommended tunings in Section 4 of this chapter, *SGB Sound Program Source List* (except for percussion instruments).

The recommended tuning values for this source list are based on an interval of C30 (See Section 7.3.6, *Interval Data*).

Also indicated for each source data item is the score data setting (interval code) for producing sounds with a C30 interval. Please refer to these settings in inputting score data.

In high and low areas, the tuning of some source data may be somewhat off. Whenever this occurs, the tuning value must be modified.

For SGB, all tunings are set 50 cents higher than the standard value (A = 440 Hz).

7.3.8 Format of Transferred Data

When Using NEWS

1. Copy **sgbt.asm** to a new transfer file, *filename*.asm.

   % cp sgbt.asm yyy.asm

   * When making transfer files, create them based on **sgbt.asm**.

2. Open **yyy.asm** and modify it as follows.

Line No.	Before Changed	After Changed
113 115	gft : 02b00h ; ; include xxx.dat	gft : yyy$, ... ; include yyy.dat . . . . .

   * When adding multiple tunes, add them beginning from line 113. Also increase the number of 'include OOO.dat' statements after line 115 by the number of tunes.

3. Execute the following command: asm700 yyy.asm.

   The above completes creation of the **yyy.hex** transfer file.

4. Convert the **yyy.hex** file completed in Step 3 to the format used by the Super NES sound generator.

   Converting to binary data:

   % cat yyy.hex | h2b -start 400 -b > yyy.bin

   Converting to hexadecimal data:

   % cat yyy.hex | h2b -start 400 > OOO.asm

When Using IS-SOUND or Original Tools

The score data file to be transferred is converted to the format used by the sound boot program.

Example:

```
dw   $0030                          ; Number of data items to transfer
dw   $2b00                          ; Transfer destination address
db   $00,$01,$02,$03,$04,$05,$06,$07  ; Score data
db   $08,$09,$0a,$0b,$0c,$0d,$0e,$0f  ; Score data
db   $00,$01,$02,$03,$04,$05,$06,$07  ; Score data
db   $08,$09,$0a,$0b,$0c,$0d,$0e,$0f  ; Score data
db   $00,$01,$02,$03,$04,$05,$06,$07  ; Score data
db   $00,$01,$02,$03,$04,$05,$06,$07  ; Score data
dw   $0000                          ; Transfer end code
dw   $0400                          ; Program start address
```

The number of data items to transfer (2 bytes) and the transfer destination address (2 bytes) are placed at the starting address of the score data. (Be careful to ensure that the data in this order.) Finally, the transfer end code (2 bytes) and the program starting address are added. (Be careful to ensure that the data is in this order.) The transfer end code is $0000.

Cautions Regarding Data Transfer

1. In SGB, the transfer destination address is $2b00, and the program starting address is $0400. Please be sure to use the correct addresses, or program control will be lost.
2. The area used for the transferred score data is approximately 8 Kbytes. A data overflow will destroy the directory.
3. If the data exceed 4 Kbytes, divide them into 2 files.
4. Transfer of score data is completely executed using system commands.

7.4 SGB Sound Program Source List

so No.	Kankichi-kun so No.	so Name	Sound Family	Envelope Type /Specific Sound	Recommended Tuning	Interval
000h	sn0	+d0.so	Sine Family	Normal envelope		
001h	sn1	+Dch.so		Envelope with extremely short decay		
002h	sn2	+d1.so		Electric keyboard envelope		
003h	sn3	+d2.so		Brass envelope		
004h	sn4	+d3.so		Pedal organ envelope		
005h	sn5	+d5.so		Banjo envelope		
006h	sn6	+d9.so		'Soft' envelope		
007h	sn7	sin.so		Normal sine wave		
008h	sn8	+d5.so	Bass Family 1	Banjo envelope	t u n, 0 1 3	
009h	sn9	+d6.so		Bass envelope	t u n, 0 1 3	
00ah	s10	+d8.so		Fretless bass envelope	t u n, 0 1 3	
00bh	s11	B1.so		Bass 1	t u n, 0 1 3	
00ch	s12	+d5.so	Bass Family 2	Banjo envelope	t u n, 0 2 0	
00dh	s13	+d6.so		Bass envelope	t u n, 0 2 0	
00eh	s14	+d9.so		'Soft' envelope	t u n, 0 2 0	
00fh	s15	B2.so		Bass 2	t u n, 0 2 0	
010h	s16	+d3.so	Guitar Family	Pedal organ envelope	t u n, 0 4 0	
011h	s17	+d5.so		Banjo envelope	t u n, 0 4 0	
012h	s18	+Dch.so		Envelope with extremely short decay	t u n, 0 4 0	
013h	s19	acg.so		Guitar	t u n, 0 4 0	
014h	s20	+d1.so	Electric Keyboard Family 1	Electric keyboard envelope		
015h	s21	+d3.so		Pedal organ envelope		
016h	s22	ep.so		Electric keyboard 1		
017h	s23	ep2.so		Electric keyboard 1	t u n, 0 0 3	C 2 0

so No.	Kankichi-kun so No.	so Name	Sound Family	Envelope Type/ Specific Sound	Recommended Tuning	Interval
018h	s24	+d1.so	Electric Keyboard Family 2	Electric keyboard envelope		
019h	s25	+d3.so		Pedal organ envelope		
01ah	s26	epf.so		Electric keyboard, soft type		
01bh	s27	pipe.so	Organ Family	Pipe organ		
01ch	s28	+d8.so	Strings Family	Fretless bass envelope	t u n, 0 8 0	C 2 0
01dh	s29	+d4.so		Strings envelope	t u n, 0 8 0	C 2 0
01eh	s30	S1.so		Strings	t u n, 0 8 0	C 2 0
01fh	s31	+d9.so	Chorus Family 1	'Soft' envelope	t u n, 1 7 0	B 0 0
020h	s32	cho1.so		Chorus 1	t u n, 1 7 0	B 0 0
021h	s33	+d3.so	Chorus Family 2	Pedal organ envelope	t u n, 1 6 5	B 1 0
022h	s34	cho2.so		Chorus 2	t u n, 1 6 5	B 1 0
023h	s35	+Dch.so	Xylophone Family	Xylophone	t u n, 0 5 5	
024h	s36	+d1.so		Electric keyboard envelope	t u n, 0 5 5	
025h	s37	+d9.so		'Soft' envelope	t u n, 0 5 5	
026h	s38	Dxlp.so		Xylophone + looping sound	t u n, 0 5 5	
027h	s39	+d1.so	Brass Family 1	Electric keyboard envelope		
028h	s40	brs.so		Brass 1		
029h	s41	brs8.so	Brass Family 2	Brass 2	t u n, 0 2 0	C 2 0
02ah	s42	+Dch.so	Trumpet Family	Envelope with extremely short decay	t u n, 0 4 0	C 2 0
02bh	s43	+d5.so		Banjo envelope	t u n, 0 4 0	C 2 0
02ch	s44	+d9.so		'Soft' envelope	t u n, 0 4 0	C 2 0
02dh	s45	tp3.so		Trumpet	t u n, 0 4 0	C 2 0
02eh	s46	+d4.so	Bassoon Family	Strings envelope		
02fh	s47	fg.so		Bassoon		
030h	s48	fl.so	Flute Family	Flute	t u n, 0 5 3	C 2 0

so No.	Kankichi-kun so No.	so Name	Sound Family	Envelope Type Specific Sounds	Recommended Tuning	Interval
031h	s49	Db.so	Percussion Instrument Family	Bass drum		
032h	s50	+Dch.so		Closed high-hat		
033h	s51	Doh.so		Open high-hat		
034h	s52	sdr3.so		Snare 1		
035h	s53	Ds.so	Percussion Family	Snare 2		
036h	s54	Dt.so		Tom (for stepping down)	t u n, 0 1 0	
037h	s55	clp.so	SE Family	Hand clap		
038h	s56	jet2.so	SE Family	Jet		
* The following (39h-3Eh) can be used with Kankichi-kun.						
039h		jet1.so	Jet			
03ah		noiz.so	Noise			
03bh		glas.so	Glass breaking			
03ch		shot.so	Shot			
03dh		river.so	River flowing			
03eh		wind.so	Wind blowing			
03fh		——	——			

Settings for source data numbers 39h-3Eh cannot be specified on Kankichi-kun. These source data can be used only with sound effects. However, they can be set using tools other than Kankichi-kun.

The shaded portions are the basic source data. The other source data items are the basic source data with modified envelopes.

The contents of the source list are also listed in the README file located in the **sobox** directory installed for NEWS.

The recommended tuning values in the source list are based on an interval of C30. (See Section 3.6.4, *Interval Data*.) With high- and low-pass filtering, the tuning of some source data may be somewhat off. Whenever this occurs, the tuning value must be modified.

The interval value is the score data setting (interval code) for producing sounds with a C30 interval. For SGB, all tunings are set 50 cents higher than the standard value (A = 440 Hz). The source data items in the empty areas do not require tuning. (In addition, they can be used without changing the interval).

7.5 Transferring Audio Data to the Score Area

In general, the score area (8 K) is provided for transferring only score data. However, audio data also can be transferred for output. Audio data can be transferred only if the following conditions are met.

- The data must not exceed the score area (8 K).
- The data is not transferred to areas other than the score area (except for the Directory and sod data).

If the data is transferred to other areas, the sound effects used by the system may no longer play or may be altered (strange sounds). Transferring data to other areas may also lead to a loss of program control. Therefore, please be certain to ensure that the above two conditions are met.

7.5.1 Required Data and Procedure for Audio Output

1. Sampling data (multiple data items permitted)

2. Score data (score used to play sampling data)

* 1 and 2 combined must occupy less than 8 Kbytes.

* The sound numbers (so No.) corresponding to the sampling data should be from among one of the following.

002h, 003h, 004h & 00Ch, 00Dh, 00Eh & 02Ah, 02Bh, 02Ch (hex No.)

Note: ***All numbers other than the above are used for system sound effects or music. Therefore, be careful to use only the above numbers.***

3. Directory and sod data corresponding to the sampling data:

* Directory and sod data are provided for each sound (so No.).

	Start Address	Data Structure	No. of Bytes
Directory	4B00h	Source start address (L)/(H) · Source loop(end)address(L)/(H)	4 bytes
sod	4C30h	so No./ adsr(1)/adsr(2)/gain/blk No. (2byte)	6 bytes

When the sound number is 000h, the directory data comprise 4 bytes beginning at 4B00h, and the sod data comprise 6 bytes beginning at 4C30h (000h cannot be used).

Please substitute the directory data and sod data values corresponding to the given sound number.

Note: ***For the sound number, however, be careful not to use any number other those shown in 2. Use of an incorrect number will cause a loss of program control.***

Transferring all of these data and issuing a BGM request will result in audio playback.

7.5.2 Transfer File Example

With sampling data consisting of a single sound with a sound number of 002h, the **Directory** data would be the 4 bytes beginning at 4B08h, and the sod data would occupy the 6 bytes beginning at 4C3Ch. In this case, ensure that the score data begin at 2B00h. Starting these data at any location other than 2B00h would cause a loss of program control. The sampling data (audio data) should be transferred to the area between 2B00h and 3AFFh.

```
dw      $0004                           ; No. of data items to transfer for Directory
dw      $4B08                           ; Directory transfer destination address
db      $00,$30,$3F,$30                 ; Directory data (4 bytes)
;
dw      $0006                           ; No. of data items to transfer for sod
dw      $4C3C                           ; Sod transfer destination address
db      $02,$FF,$E0,$B8,$02,$B0         ; Sod data (6 bytes)
;
dw      $0020                           ; No. of score data items to transfer
dw      $2B00                           ; Score data transfer destination address
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Score data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Score data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Score data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Score data
;
dw      $0040                           ; No. of sampling data items to transfer
dw      $3000                           ; Sampling data transfer destination address
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Sampling data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Sampling data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Sampling data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Sampling data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Sampling data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Sampling data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Sampling data
db      $00,$01,$02,$03,$04,$05,&06,$07 ; Sampling data
;
dw      $0000                           ; Transfer end code
dw      $0400                           ; Program start address
```

When using multiple sampling data items, also transfer the Directory and sod data specified for each item in Step 2.

Note: ***Be careful not to rewrite the*** *Directory* ***and*** *sod* ***data used by the system.***

THIS PAGE WAS INTENTIONALLY LEFT BLANK

CHAPTER 8 GAME BOY MEMORY CONTROLLERS (MBC) 233

8.1 MBC1 233

8.1.1 Overview 233
8.1.2 Description of Registers 233
8.1.3 Memory Map 235

8.2 MBC2 236

8.2.1 Overview 236
8.2.2 Description of Registers 236
8.2.3 Memory Map 236
8.2.4 Backup RAM 236

8.3 MBC3 237

8.3.1 Overview 237
8.3.2 Description of Registers 237
8.3.3 Accessing the Clock Counters 238
8.3.4 Memory Map 239
8.3.5 Programming Cautions 240

8.4 MBC5 242

8.4.1 Overview 242
8.4.2 Registers 242
8.4.3 Memory Map 242
8.4.4 Description of Registers 243
8.4.5 Programming Cautions 244
8.4.6 Examples of MBC5 programs on DMG and CGB 245

8.5 MBC5 (with rumble feature) 246

8.5.1 Overview 246
8.5.2 Registers 246
8.5.3 Memory Map 247
8.5.4 Description of Registers 248
8.5.5 Motor Control 249
8.5.6 Programming Cautions 250
8.5.7 Physical Effects of Vibration on the Body 251

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 8 GAME BOY MEMORY CONTROLLERS (MBC)

8.1 MBC1

8.1.1 Overview

MBC1 is a memory controller that enables the use of 512 Kbits (64 Kbytes) or more of ROM and 256 Kbits (32 Kbytes) of RAM. It can be used as follows.

- To control up to 4 Mbits of ROM
  When used to control up to 4 Mbits (512 Kbytes) of ROM, MBC1 can control up to 256 Kbits (32 Kbytes) of RAM.

- To control 8 Mbits or more of ROM
  When MBC1 is used to control up to 8 MBits (1 MB) or 16 MBits (2 MB) of ROM, the following conditions apply

  - When used to control 8 MBits of ROM
    MCB cannot use ROM addresses 080000h-083FFFh (Bank 20h)

  - When used to control 16 MBits of ROM

    MBC1 cannot use ROM Addresses
    - 080000h-083FFFh (Bank 20h)
    - 100000h-103FFFh (Bank 40h)
    - 180000h-183FFFh (Bank 60h)

    RAM use by MBC1 is restricted to 64 Kbits (8 Kbytes).

8.1.2 Description of Registers

- Register 0: RAMCS gate data (serves as write-protection for RAM)
  Write addresses: 0000h-1FFFh Write data: 0Ah
  Writing 0Ah to 0h-1FFFh causes the $\overline{CS}$ to be output, allowing access to RAM.

- Register 1: ROM bank code
  Write addresses: 2000h-3FFFh Write data: 01h-1Fh
  The ROM bank can be selected.

- Register 2: Upper ROM bank code when using 8 Mbits or more of ROM (and register 3 is 0)
  Write addresses: 4000h-5FFFh Write data: 0-3
  The upper ROM banks can be selected in 512-Kbyte increments.
  Write value of 0 selects banks 01h-1Fh
  Write value of 1 selects banks 21h-3Fh
  Write value of 2 selects banks 41h-5Fh
  Write value of 3 selects banks 61h-7Fh

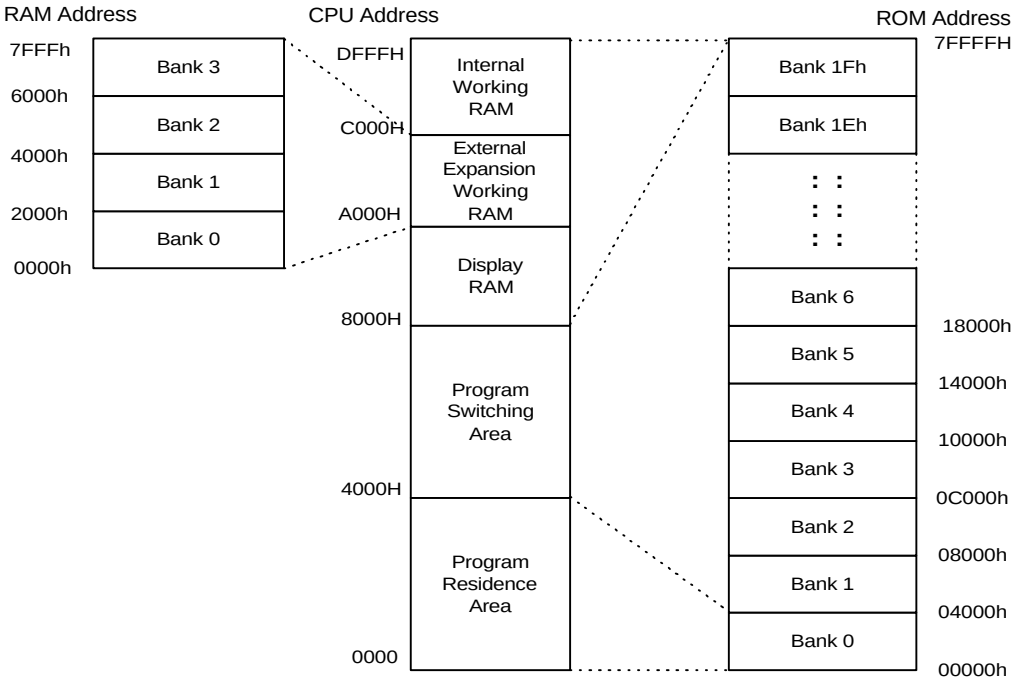
  : RAM bank code when using 256 Kbits of RAM (and register 3 is 1)

  Write addresses: 4000h-5FFFh Write data: 0-3
  The RAM bank can be selected in 8-Kbyte increments.

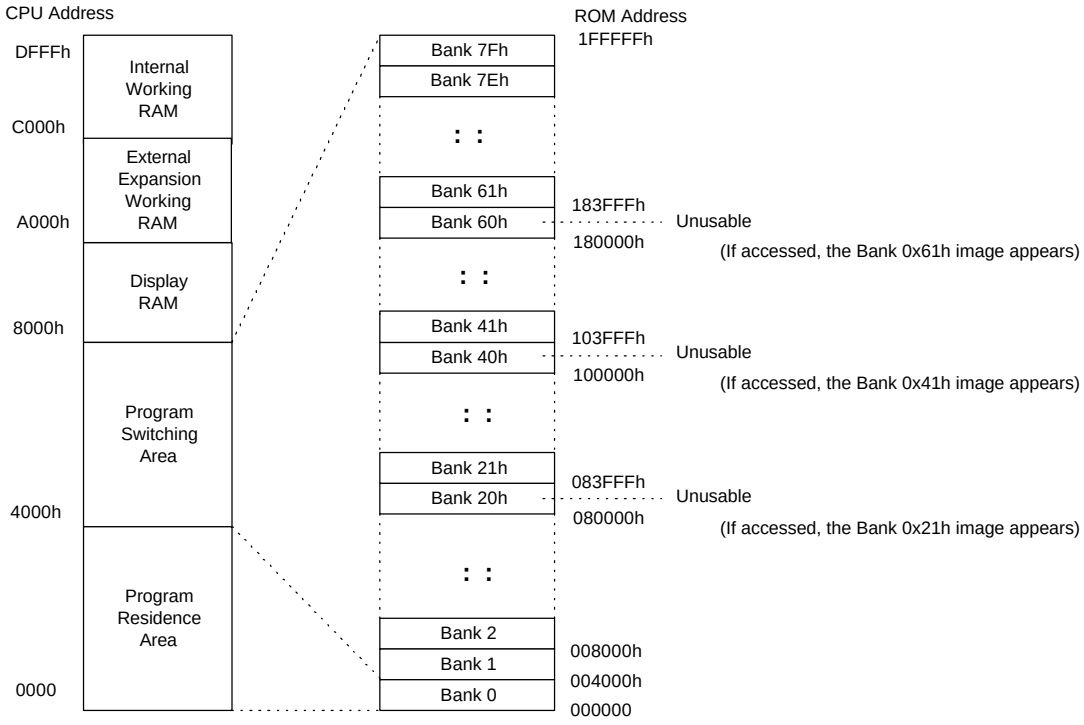
- Register 3: ROM/RAM change
  Write addresses: 6000h-7FFFh Write Data: 0-1
  Writing 0 causes the register 2 output to control switching of the higher ROM bank.
  Writing 1 causes the register 2 output to control switching of the RAM bank.

8.1.3 Memory Map

- When Used to Control up to 4 Mbits of ROM



- When Used to Control up to 8 Mbits of ROM



8.2 MBC2

8.2.1 Overview

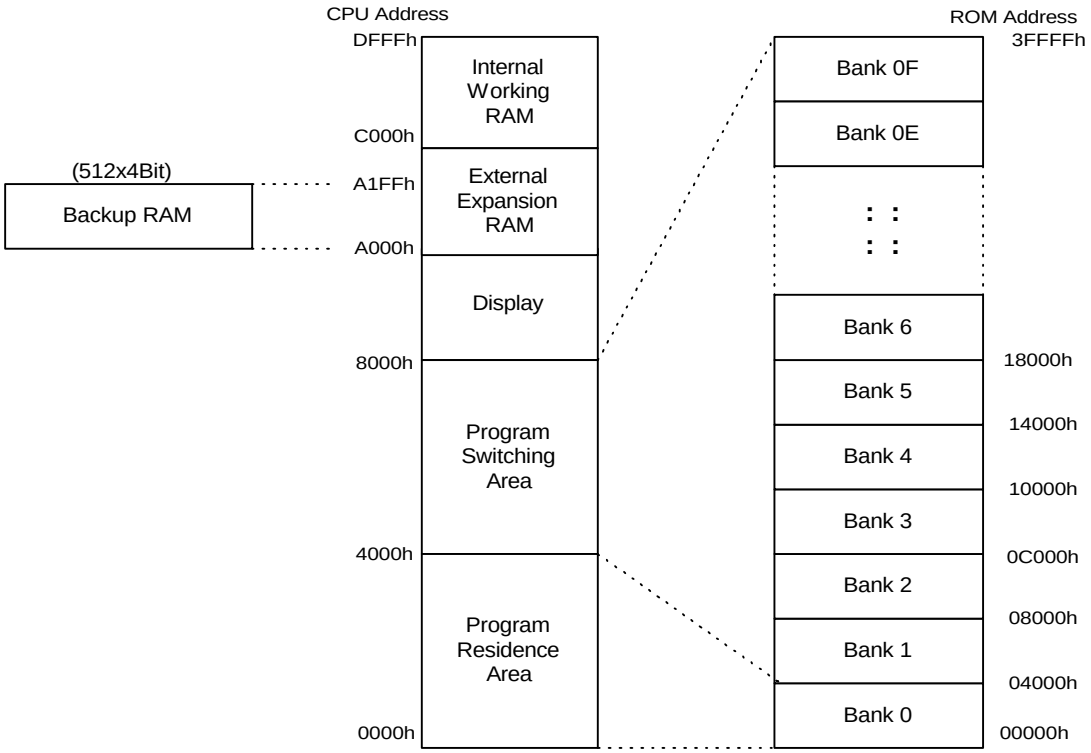
Controller for up to 2 Mbits (256 Kbytes) of ROM with built-in backup RAM (512 x 4 bits).

8.2.2 Description of Registers

- Register 0: RAMCS gate data (serves as write-protection for RAM)
  Write addresses: 000h-0FFh Write data: 0Ah
  Writing 0Ah to 0000-0FFFh causes the $\overline{CS}$ to be output, allowing access to RAM.

- Register 1: ROM bank code
  Write addresses: 2100h-21FFh Write data: 01h-0Fh
  The ROM bank can be selected.

8.2.3 Memory Map



8.2.4 Backup RAM

Allocated to the D0-D3 areas of CPU addresses A000h-A1FFh
Backup RAM is write-protected by a power-on reset.
To protect backup data, avoid removing write protection unless necessary.

8.3 MBC3

8.3.1 Overview

MBC3 is the memory bank controller that allows use of between 512 Kbits (64 Kbytes) and 16 Mbits (2 MB) of ROM and 256 Kbits (32 Kbytes) of RAM.
Built into the controller are clock counters that operate by means of an external crystal oscillator (32.768 KHz). The clock counters are accessed by RAM bank switching.
RAM and clock counter data can be backed up by an external lithium battery.

8.3.2 Description of Registers

Settings for control registers 0-3 are specified by writing data to the ROM area.

- Register 0: Write protects RAM and the clock counters (default: 0)
  Write addresses: 0000h-1FFFh Write data: 0Ah
  Allows access to RAM and the clock counter registers.

- Register 1: ROM bank code (default: 0, selects ROM bank 1)
  Write addresses: 2000h-3FFFh Write data: 01h-7Fh
  Allows the ROM bank to be selected in 16-Kbyte increments.

- Register 2: RAM bank code (default: 0, selects RAM bank 0)
  Write addresses: 4000h-5FFFh Write data: 0-3
  Allows the RAM bank to be selected in 8-Kbyte increments.

  Write addresses: 4000h-5FFFh Write data: 08h-0Ch
  Allows a clock counter to be selected.

Data	Register	Range of Values	Function
08h	RTC_S	0-59 (0-3Bh)	Seconds counter (6 bits)
09h	RTC_M	0-59 (0-3Bh)	Minutes counter (6 bits)
0Ah	RTC_H	0-23 (0-17h)	Hours counter (5 bits)
0Bh	RTC_DL	0-255 (0-FFh)	Lower-order 8 bits of days counter
0Ch	RTC_DH	bit7 bit0 [ ] [ ] 0 0 0 0 0 [ ] Bit 0: Most significant bit of days counter Bit 6: HALT Bit 7: Carry bit of days counter	Higher-order bit and carry bit of days counter. HALT starts and stops the clock counters.

* The days counter consists of a 9-bit counter + a carry bit. Thus, it can count from 0 to 511 (000h-1FFh).
* Once the carry bit is set to 1, it remains 1 until 0 is written.
* The counters operate when HALT is 0 and stop when HALT is 1.
* Values outside the given counter ranges will not be correctly written.

- Register 3: Latches the data for all clock counters (default: 0)
  Write addresses: 6000h-7FFFh Write Data: 0 → 1
  Writing 0 → 1 causes all counter data to be latched. The latched contents are retained until 0 → 1 is written again.

8.3.3 Accessing the Clock Counters

The clock counter registers are assigned to the external expansion RAM area of the CPU address space. To access the clock counters, RAM bank switching must first be performed.

External expansion RAM Area (A000h-BFFFh) Bank Map

Bank	Device	Notes
00h	RAM BANK 0	
01h	RAM BANK 1	
02h	RAM BANK 2	
03h	RAM BANK 3	
:		Not used
08h	Seconds counter	
09h	Minutes counter	
0Ah	Hours counter	
0Bh	Days counter (L)	
0Ch	Days counter (H)	
:		Not used

The following are examples of accessing the clock counters.

8.3.3.1 Reading

The clock counters are accessed by first writing 0x0A to register 0. This opens the gate used to access the counters. To read clock counter values, write 1 to register 3 to latch the values of all the registers. If the value of register 3 is already 1, first set it to 0 and then to 1. While this register is set to 1, the clock counters will operate but the latched values of all of the clock counters will not change. This allows the clock counters to be read.

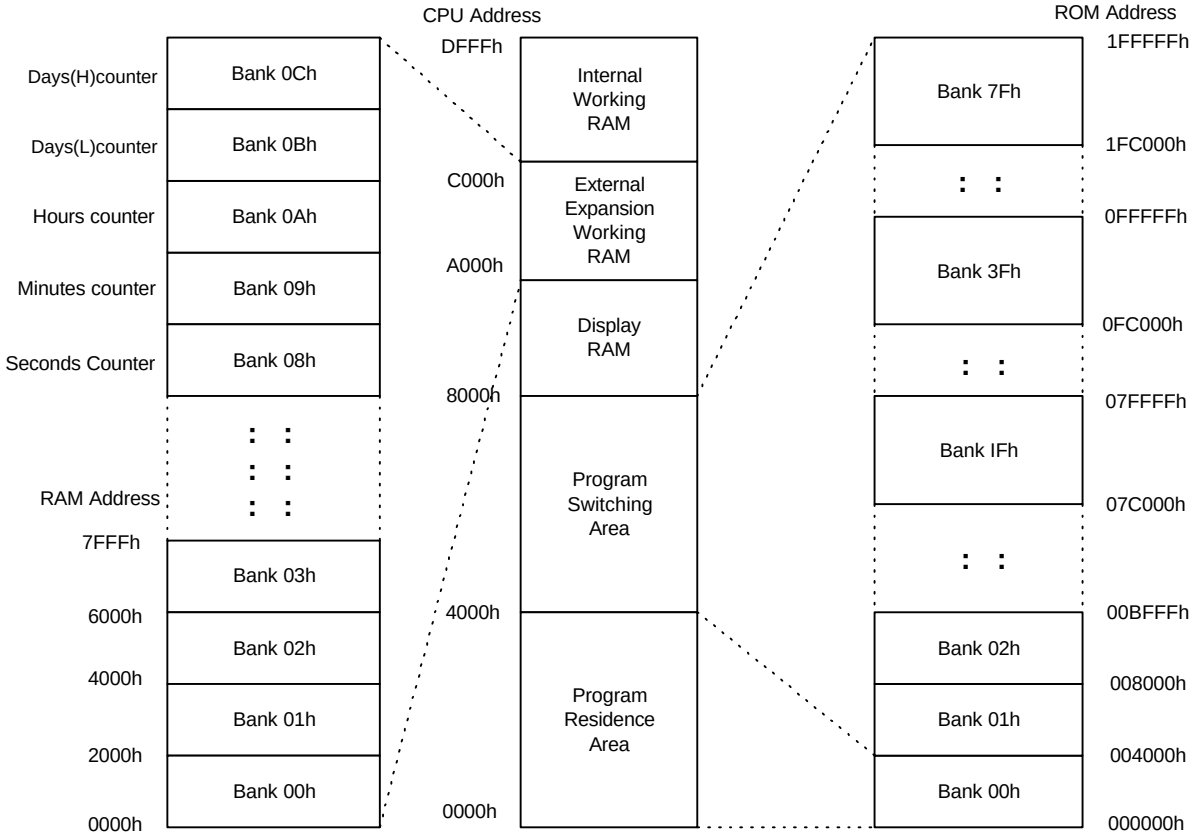
For example, the seconds counter register can be accessed and read by first setting the RAM bank to 8, then reading from any CPU address between A000h and BFFFh.

8.3.3.2 Writing

Writing 0Ah to register 0 opens the access gate, allowing each clock counter register to be written to.

8.3.4 Memory Map

- ROM bank 0 is assigned to the program residence area (0000h-3FFFh) of the CPU memory space (fixed).
- One bank from among ROM banks 01h-7Fh can be assigned to the program switching area (4000h-7FFFh) of the CPU memory space.
- One bank from among RAM banks 0-3 and the clock counter registers (RAM banks 08h-0Ch) can be assigned to the external expansion working RAM area (A000h-BFFFh) of the CPU memory space.



DMG-06-4216-001-D
Released: 10/10/2000

8.3.5 Programming Cautions

8.3.5.1 Accessing the Clock Counters

Although counting up of the clock counters themselves and accessing the clock counters from the CPU are performed asynchronously, clock counter failure may result if both operations are performed at the same time. To prevent this, MBC3 provides an interface circuit for WR signals from the CPU. Use of this circuit necessitates a delay when accessing control register 3 and the clock counter registers (RTC_S, RTC_M, RTC_H, RTC_DL, and RTC_DH). Thus, whenever accessing these registers consecutively, interpose a delay of 4 cycles between accesses.

When reading clock counter data:

- Latch all clock counter data using control register 3.

4-cycle delay required

- Read the data in the clock counter registers.

When writing values to the clock counters:

- Set data in clock counter register RTC_S.

4-cycle delay required

- Set data in clock counter register RTC_M.

4-cycle delay required

- Set data in clock counter register RTC_H.

4-cycle delay required

- Set data in clock counter register RTC_DL.

4-cycle delay required

- Set data in clock counter register RTC_DH.

8.3.5.2 Condensation

MBC3 uses a crystal oscillator for its clock counter operation, and condensation on the oscillator may halt its oscillation, preventing the clocks from counting up. Once the condensation disappears, the clocks will resume counting up from where they stopped. However, please ensure that the counter stoppage does not result in a loss of program control.

8.3.5.3 Control Register Initialization

Although control registers 0-3 are initialized (see Section 3.2, *Description of Registers*) when Game Boy power is turned on, they are not initialized by a hard reset of SNES when Super Game Boy is used. Therefore, please be sure to implement a software reset of these registers.

8.3.5.4 Clock Counter Registers

When commercial Game Boy software that uses MBC3 is shipped from the factory, the values of the clock counter registers are undefined. Therefore, please ensure that these registers are initialized.

8.4 MBC5

8.4.1 Overview

Supports CGB double-speed mode.

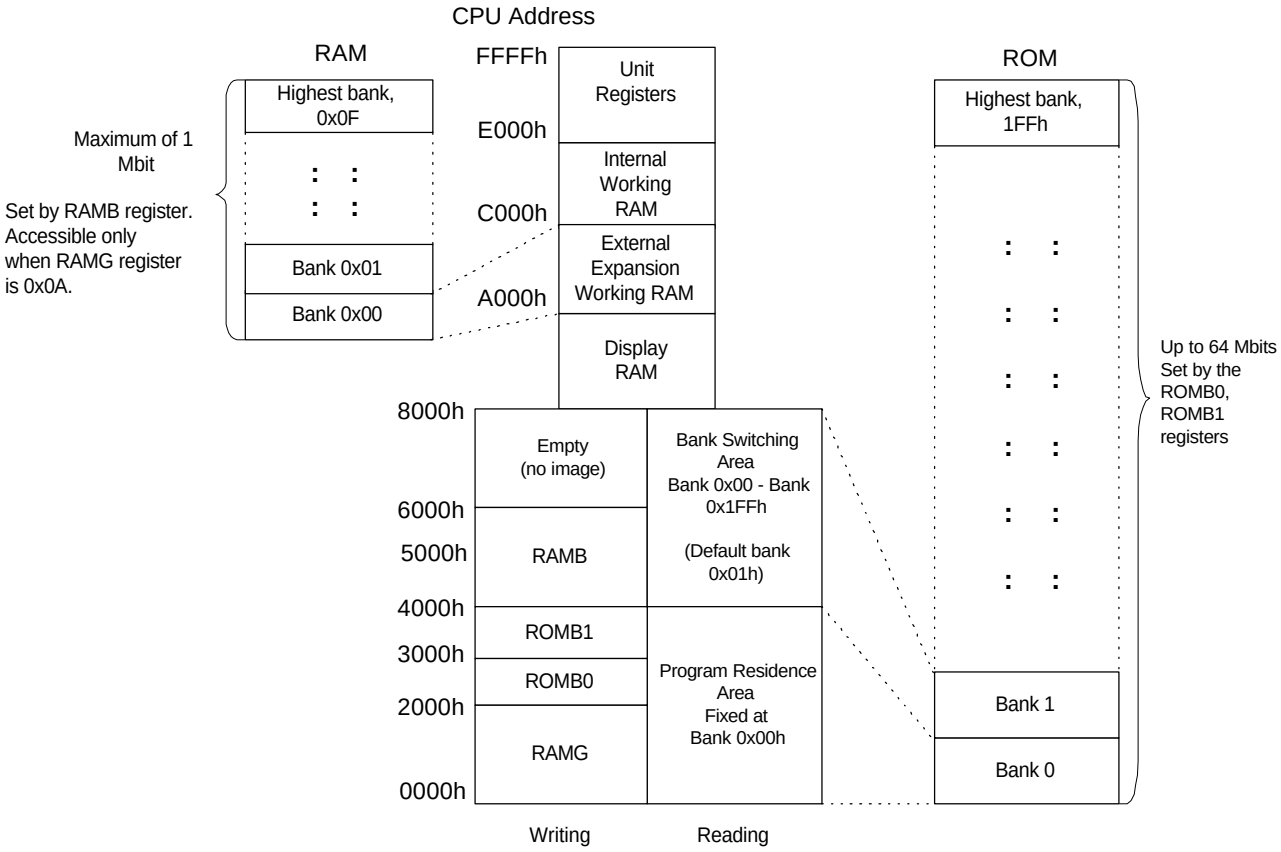
MBC5 can use up to 64 Mbits of ROM (512 banks of 128 bits each) and 1 Mbit of RAM (16 banks of 64 Kbits each).

Upwardly compatible with MBC1.

8.4.2 Registers

Name	Addresses (hex)
RAMG	0000-1FFF
ROMB0	2000-2FFF
ROMB1	3000-3FFF
RAMB	4000-5FFF

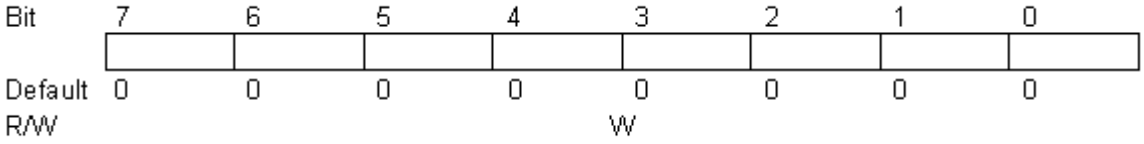
8.4.3 Memory Map



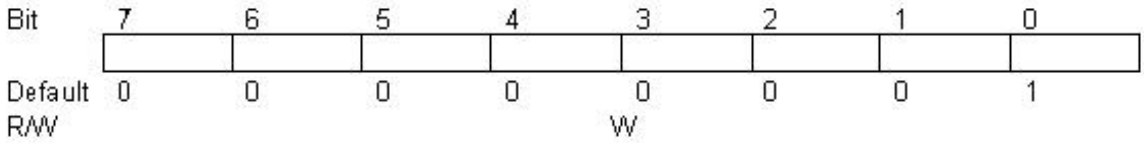
During a write, data is written to the bank control registers at CPU addresses 0000h-7FFFh. During a read, the contents of ROM are read from these addresses.

8.4.4 Description of Registers

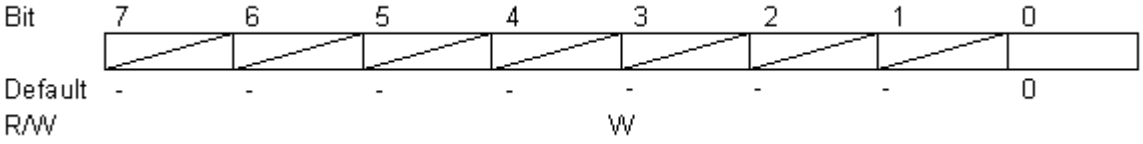
- Register for Specifying External Expansion Memory (RAMG)
  Specifies whether external expansion RAM is accessible. Access to this RAM is enabled by writing 0Ah to the RAMG register space, 0000h-1FFFh. Writing any other value to this register disables reading to and writing from RAM.



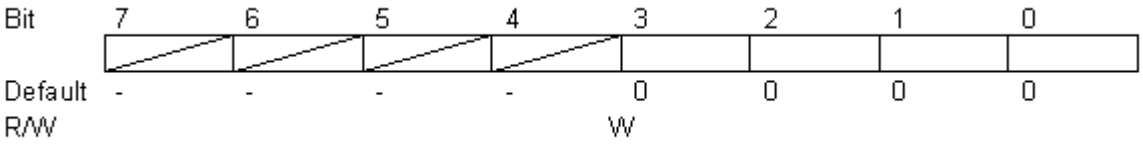
- Lower ROM Bank Register (ROMB0)
  Specifies the lower-order 8 bits of a 9-bit ROM bank.
  The ROM bank can be changed by writing the desired ROM bank number to the ROMB0 register area, 2000h-2FFFh.



- Upper ROM Bank Register (ROMB1)
  Specifies the higher-order 1 bit of a 9-bit ROM bank.
  The ROM bank can be changed by writing the desired ROM bank number to the ROMB1 register area, 3000h-3FFFh.



- RAM Bank Register (RAMB)
  Specifies the RAM bank
  The RAM bank can be changed by writing the desired RAM bank number to the RAMB register area, 4000h-5FFFh.



Note	***Although the bits marked with ▨ are ignored by MBC5, they should be used after being set to 0. The default values are set automatically when power is turned on.***

8.4.5 Programming Cautions

8.4.5.1 When Migrating from MBC1 to MBC5

- Use of Register 1
  If an MBC1 program uses register 1 (ROM bank control register) addresses 3000h-3FFFh, the bank intended for selection by ROMB1 in MBC5 will not be selected.
  Addresses 2000h-2FFFh of register 1 should be used by programs that use MBC1.

- Use of Register 2
  Note that in MBC1, programs that use 8 Mbits or more use register 2 (ROM or RAM bank control register) for the high ROM bank. Consequently, in MBC5 the RAM bank is different while the ROM bank is unchanged.

- ROM Banks 20h, 40h, and 60h
  ROM banks 20h, 40h, and 60h cannot be used in MBC1, but they can be used in MBC5.

- MBC1 Register 3 (ROM/RAM change)
  Because the addresses of ROM and RAM are independent of each other in MBC5, ROM/RAM switching is unnecessary.
  Any write instructions to register3 left in a program that uses MBC1 are ignored by MBC5 and have no effect.

8.4.5.2 General Notes

- Memory Image
  If a memory device is used that uses less than the maximum amount of memory available (ROM: 64 Mbits; RAM: 1 Mbit) , a memory image is generated for the empty bank area. Therefore, please do not develop software that uses an image, because it may cause failures.

- RAM Data Protection
  To protect RAM data, it is recommended that RAM access be disabled when RAM is not being accessed (RAMG ← 00h) .

- Specifying External Sound Input (VIN)
  Always use the sound control register (NR50) with bits 7 and 3 (VIN function OFF) set to 0. Because the VIN terminal is used in development flash ROM cartridges, using the register with VIN set to ON will produce sound abnormalities.

8.4.6 Examples of MBC5 programs on DMG and CGB

- Set the bank switching area (4000h-7FFFh) to 1FFh.

```
LD A,$FF
LD ($2000),A      ;ROMB0 setting
LD A,$01
LD ($3000),A      ;ROMB1 setting
|
|
|
```

- Set the external expansion memory area (A000h-BFFFh) to 0Fh.

```
LD A,$0F
LD ($4000),A      ; RAMB setting
LD A,$0A
LD ($0000),A      ; Enable access to RAM
|          }
|          } RAM Access Processing
|          }
LD A,$00
LD ($0000), A     ; Disable access to RAM
```

8.5 MBC5 (WITH RUMBLE FEATURE)

8.5.1 Overview

This cartridge is the same as the previous MBC5 cartridge but also includes a rumble motor and size AAA battery to power the motor. The motor is controlled by the program using the MBC5 RAM bank register (RAMB, bit 3).

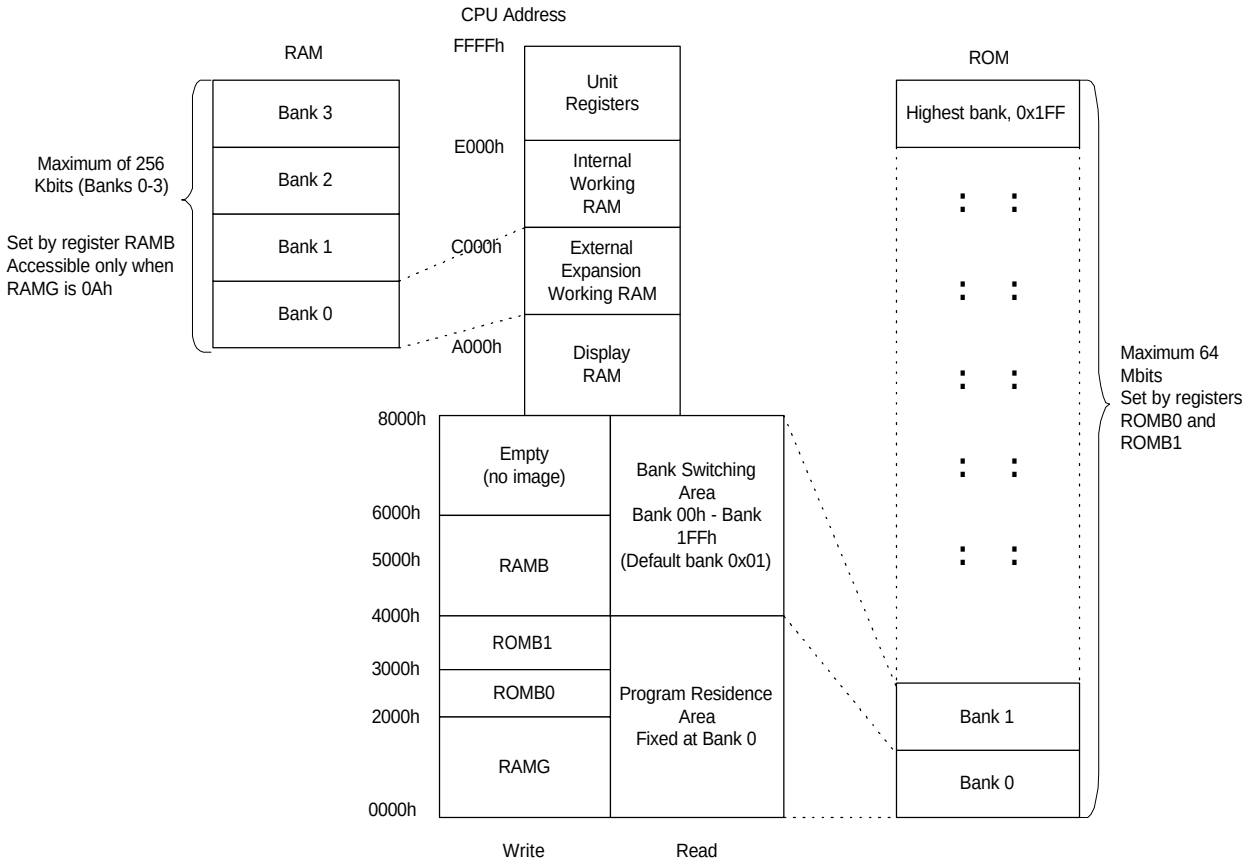
MBC5 supports CGB normal and double-speed modes.

Up to 64 Mbits (512 banks of 128 Kbits each) of ROM and 256 Kbits of RAM (4 banks of 64 Kbits each) can be used.

8.5.2 Registers

Name	Addresses (hex)	Notes
RAMG	0000-1FFF	Each register executes its control using any one of the address spaces at left.
ROMB0	2000-2FFF	
ROMB1	3000-3FFF	
RAMB	4000-5FFF	

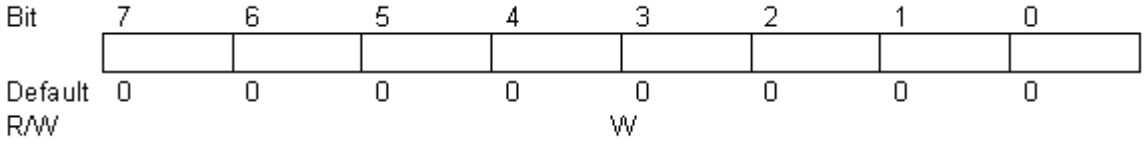
8.5.3 Memory Map



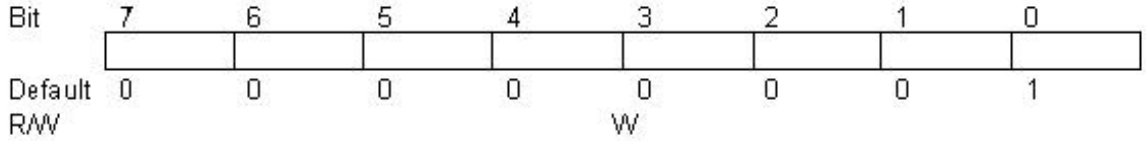
* During a write, data is written to the bank control registers at CPU addresses 0000h-7FFFh. During a read, the contents of ROM are read from these addresses.

8.5.4 Description of Registers

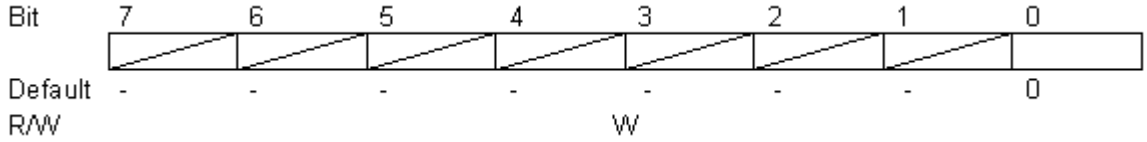
- Register for Specifying External Expansion Memory (RAMG)
  Specifies whether external expansion RAM is accessible. Access to this RAM is enabled by writing 0Ah to the RAMG register (any single address in 0000h-1FFFh). Writing any other value to this register disables reading to and writing from RAM.



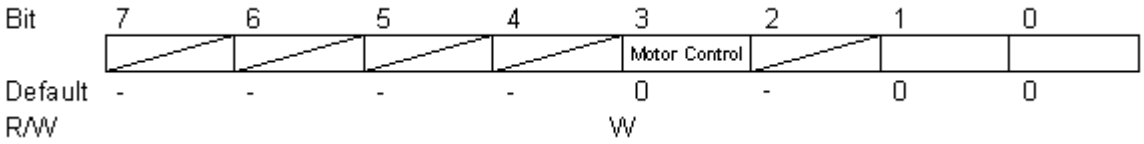
- Lower ROM Bank Register (ROMB0)
  Specifies the lower-order 8 bits of a 9-bit ROM bank.
  The ROM bank can be changed by writing the desired ROM bank number to the ROMB0 register (any single address in 2000h-2FFFh).



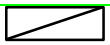
- Upper ROM Bank Register (ROMB1)
  Specifies the higher-order 1 bit of a 9-bit ROM bank.
  The ROM bank can be changed by writing the desired ROM bank number to the ROMB1 register (any single address in 3000h-3FFFh).



- RAM Bank Register (RAMB)
  Specifies the RAM bank.
  The RAM bank can be changed by writing the desired RAM bank number to the RAMB register (any single address in 4000h-5FFFh).



Bits 0-1: Register for RAM bank setting
Bit 3: Motor control register (1: motor ON; 0: motor OFF)

Note: ***Be sure to set the bits marked with***  ***to 0 before using them. The default values are set automatically when power is turned on.***

8.5.5 Motor Control

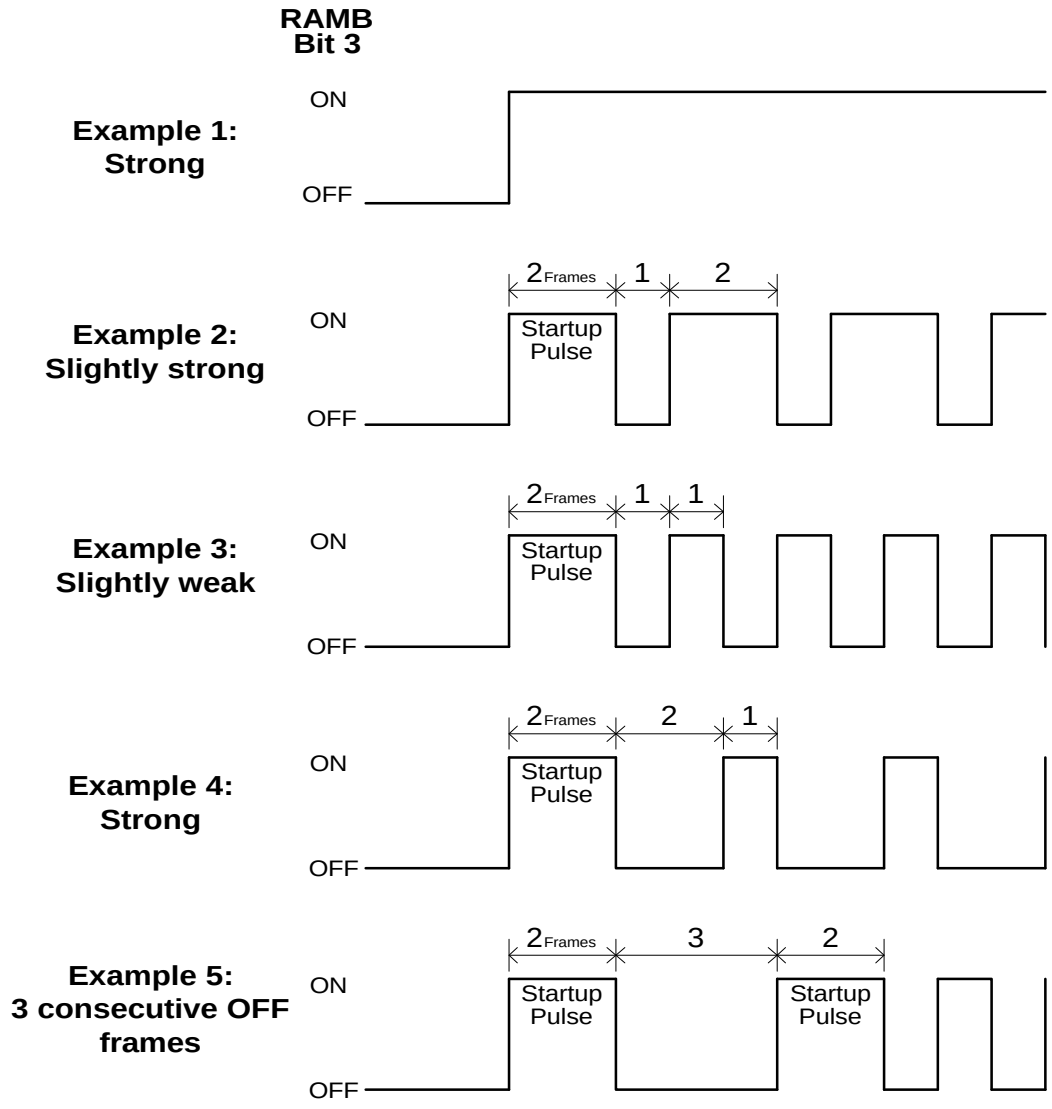
8.5.5.1 Vibration Level

Control of the rumble motor consists of setting it to ON or OFF.
The vibration level can be controlled by sending pulses of combined ON/OFF instructions in short cycles. Please comply with the following points when implementing vibration control.

(1) Set the frame rate to 1 frame per 1/60 second and control vibration frame by frame.

(2) At the start of vibration control, send a startup pulse (at least 2 ON frames).
A startup pulse also should be sent if the width of an OFF pulse is 3 or more consecutive frames. This is necessary because startup from a complete stop requires a certain amount of time. (see Example 5 below).

8.5.5.2 Vibration Pulse Examples



DMG-06-4216-001-D
Released: 10/10/2000

8.5.6 Programming Cautions

IMPORTANT

8.5.6.1 Memory Image

If a memory device is used that uses less than the maximum amount of memory available (ROM: 64 Mbits; RAM: 256 Kbits) , an empty bank area (memory image) results. Please do not access this empty bank area. Doing so may result in faulty operation.

8.5.6.2 RAM Data Protection

To protect RAM data, it is recommended that RAM access be disabled (RAMG←00h) when RAM is not being accessed.

8.5.6.3 Specifying External Sound Input (VIN)

Always use the sound control register (NR50) with bits 7 and 3 set to 0 (VIN function OFF). Because the VIN terminal is used in development flash ROM cartridges, using the register with VIN set to ON will produce sound abnormalities.

8.5.6.4 Disabling Vibration Using the SGB, SGB2, or N64Transfer Pak

When MBC5 is used by SGB, SGB2, or the N64 Transfer Pak, vibration should be turned off by the program to prevent failures caused by a faulty connection. For methods of recognizing SGB and SGB2, see the description of the MLT_REQ command in Chapter 6, Section 3.2, "System Command Details. With the N64 Transfer Pak, vibration is controlled by the N64 software. Therefore, N64 software programs that support MBC5 should not write data to bit 3 of the RAM bank register.

8.5.6.5 Limiting the Period of Continuous Vibration

To prevent physical effects in the user such as numbness as a result of continuous vibration, limit the duration of continuous vibration as indicated below, regardless of the vibration strength (see Section 8.5.5.2, *Vibration Pulse Examples*).

- The duration of continuous vibration should generally be limited to a maximum of 1 minute.
- The period of no vibration between the finish of one period of vibration and the start of the next period generally must be at least as long as the vibration time.

The above points are guidelines that should be followed in most cases. However, if adhering to these guidelines is made difficult by factors such as the game content, take appropriate measures while keeping in mind the points noted in Section 8.5.7, *Effects of Vibration on the Body*.

8.5.6.6 Disabling Vibration for Resets and Pauses

Vibration should be halted during resets and pauses.

When power is turned on, the unit should not be vibrated until some input is received from the controller.

8.5.6.7 Rumble Feature Selection

The user should be allowed to set the rumble feature to ON or OFF or to select strong, mild, or OFF by means such as an initial-settings screen at the start of the game. In addition, the program should allow the user to easily change these settings even during a game if, for example, they are uncomfortable with the vibration. Such changes also should be allowed a pause.

8.5.6.8 Changes in Vibration Level with Battery Use

If the battery that powers the motor (Size AAA alkaline battery) wears out, the perceived vibration level will be reduced even if the requested vibration level remains the same. Therefore, rumble operation should be checked both when the battery is new (1.6 V) and when it is at the end of its life (1.1 V).

8.5.7 Physical Effects of Vibration on the Body

Users have occasionally experienced numbness for some time after continuous vibration lasting several tens of seconds to several minutes. This may occur regardless of the strength of the vibration (see Section 8.5.5.2, *Vibration Pulse Examples*).

Unfortunately, the effects of continuous vibration on the body are not yet clear. Thus, the guidelines presented in Section 8.5.6.5, *Limiting the Period of Continuous Vibration*, are intended to give priority to user safety. However, software development requires free thinking and original ideas, and there may well be cases in which the use of continuous vibration in a game is desirable.

Because each game is different, the limitations presented in Section 8.5.6.5 are by their nature not restrictions that should be enforced digitally. It is instead preferable for the developer to adequately consider user safety when determining the game's content.

For example, even supposing that continuous vibration does last for more than 1 minute, it may not pose a safety problem if it is used infrequently, such as only when special events occur. Conversely, if vibrations lasting several seconds to several tens of seconds are repeated at short intervals, the effects on the user may be the same as with continuous, long-term vibration.

Thus, the guidelines presented in Section 8.5.6.5 are not absolute restrictions. However, even if a program varies from these guidelines, the following points should be considered minimum requirements and strictly observed.

- Continuous vibration should not exceed 3 minutes for any reason.

- Because the effects of continous vibration vary from person to person, the strength of these effects on the user should not be determined independently by the developer. Rather, this determination should be arrived at after considering the opinions of many others during debugging and other phases of development.

CHAPTER 9 POCKET PRINTER 255

9.1 Overview 255

9.2 Communication Specifications 255

9.2.1 Bidirectional Communication 255
9.2.2 Transfer Interval For Each Byte 255
9.2.3 Packets and the Transfer Interval 255
9.2.4 Synchronism Check when Connecting 255

9.3 Communication Data Definitions 256

9.3.1 Transferring to the Printer 256
9.3.2 Receiving from the Printer 257
9.3.3 Handling Errors 257

9.4 Packet Details 259

9.4.1 The Initialization and Connection Packet 259
9.4.2 Print Instruction Packet 259
9.4.3 Data Packet 261
9.4.4 Data-End Packet 262
9.4.5 Break Packet 262
9.4.6 NUL Packet 262
9.4.7 Packet Error 262
9.4.8 Other Packets 262

9.5 Printer Status and Packets 263

9.6 Printer Print Sequence 264

9.7 Processing of Connection Evaluation and Preamble Detection Failure 265

9.7.1 Connection Evaluation (includes cable disconnection) 265
9.7.2 Preamble Detection Failure 266

9.8 Print Data 267

9.9 Compression Algorithm 268

9.10 Hardware Specifications 269

9.10.1 General Specifications 269
9.10.2 Dimensions and Weight 269

9.11 Miscellaneous 269

9.11.1 Cautions when Debugging 269
9.11.2 Sample Programs Provided by Nintendo (subroutines) 269

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

CHAPTER 9 POCKET PRINTER

9.1 Overview

These specifications define the serial protocol used to send print and control data from Game Boy to the Pocket Printer (abbreviated to printer). Game Boy sends data to the printer in packets, and the printer responds by returning 2 bytes of status information.

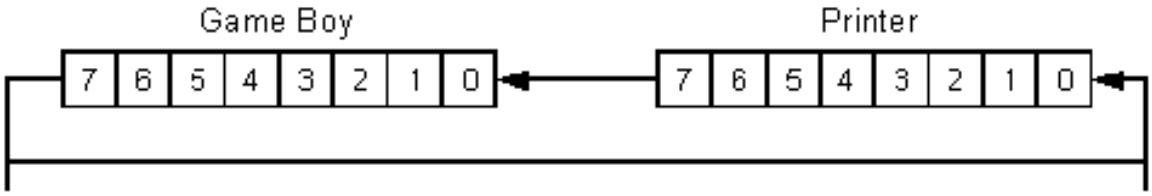
9.2 Communication Specifications

9.2.1 Bidirectional Communication

Serial transfers between Game Boy and the printer are performed in the Game Boy specification communication format (bidirectional).

The shift clock is furnished by the Game Boy. Both Game Boy and the printer start transmission from the most significant bit (MSB).

For more information , see Chapter 1, Section 2.5.1, *Serial Cable Communication*.



9.2.2 Transfer Interval For Each Byte

An interval of 270 μs to 5 ms must be interposed between each byte sent. Thus, care should be exercised regarding factors like interrupts when programming.

9.2.3 Packets and the Transfer Interval

Each type of data sent by the Game Boy is sent in a packet. An interval of 270 μs to 117 ms must be allowed between the transfer of each packet. Thus, care should be exercised regarding factors like interrupts when programming.

9.2.4 Synchronism Check when Connecting

After the connection between the Game Boy and printer is confirmed, the Game Boy sends a NUL packet every 100 msec for a synchronism check of the connection. If the Game Boy determines that a connection is unnecessary and does not send a NUL packet in the prescribed time, the printer will determine that the connection is abnormal and will wait in an initialized state for a signal from the Game Boy.

9.3 Communication Data Definitions

This section defines the following data items (packet types and data) by function.

9.3.1 Transferring to the Printer

The packet types are as follows.

Packet Type	Code
Initialization and connection packet	01
Print instruction packet	02
Data packet	04
Data end packet	04
Break packet	08
NUL packet	0F

Each of the above packet types is in the following format.

Preamble	Header	Data	Checksum	Dummy

Preamble: 2 bytes of data: 88h x 1 + 33h x 1. Abbreviated PA below.
Header: 4 bytes of contiguous data that represent the following.
- Byte 1: Packet type
  - 01: Initialization and connection packet
  - 02: Print instruction packet
  - 04: Data packet
  - 08: Break packet
  - 0F: NUL packet
- Byte 2: In the case of a data packet, indicates compression/no compression.
  If another type of packet, fixed at 0x00.
- Bytes 3 and 4: Data volume (2 bytes), number of bytes of data

Data: Data in Game Boy character data format. Print instruction data.
Checksum: 2 bytes of data representing the sum of the header + all data in the data portion of the packet.
Dummy: 2 bytes of dummy data used to obtain status information from the printer.
In the data received from the printer in place of the dummy data, byte 1 holds the peripheral device number and byte 2 holds the printer status.

9.3.2 Receiving from the Printer

The printer returns 2 bytes of status data.

Byte 1: Device number

1	0	0	0	0	0	0	1

The value of the MSB is always 1. The lower-order 7 bits represent the device number. The Pocket Printer is device number 1.

Byte 2: Status

LowBat	ER2	ER1	ER0	Untran	Full	Busy	Sum

LowBat: 1 = Low-battery error bit
0 = Battery OK
ER2: 1 = Other error
ER1: 1 = Paper jam (abnormal motor operation)
ER0: 1 = Packet error
Untran: 1 = Unprocessed data present
0 = No unprocessed data present
Full: 1 = Image data full
0 = Image data not full
Busy: 1 = Printer busy
0 = Printer ready
Sum: 1 = Checksum error
0 = Data OK

Status information is sent in reply to each 2 bytes of dummy data sent by the Game Boy.

* The status returned by the printer is FF FF when the printer is not connected to the Game Boy or not powered on.

9.3.3 Handling Errors

Either an error number listed below or the error number plus a description of the error would be sent to the display screen in response to an error flag in byte 2. (This information is also presented in the user's manual of the Pocket Printer. That information must be used together with the information given here.)

Status: Byte 2	Error No.
low Bat = 1	01
FF FF	02
ER1 = 1	03
ER2 = 1	04

* Error No. 02 is represented using both status bytes.

ER0 = 1 likely indicates program failure.

When an error is generated, always sever communication with the printer and inform the user of the type of error.

A value other than 81h for the first status byte means that a device other than the Pocket Printer is connected. This should be conveyed to the user as an error message.

9.4 Packet Details

9.4.1 The Initialization and Connection Packet

This packet is used to initialize the printer and check the connection. If the Game Boy sends a packet for checking the printer connection and a printer is connected, it returns a 2-byte status code and initializes for the start of print processing. This packet must always be sent when the Game Boy starts to access the printer. It allows transferred data to be invalidated (reset).

Actual Data

88	33	01	00	00	00	01	00	00	00
PA		Header				Checksum		Dummy	

This packet has no data section.

Normal status: 81h and 00h (For more information, see Section 3, *Communication Data Definitions*.)

Not connected: FFh and FFh.

9.4.2 Print Instruction Packet

Used for print instructions for single-sheet mode and copy mode (for specifying the number of sheets).

Example:

88	33	02	00	04	00	01	13	E4	40	3D	01	00	00
PA		Header				Data				Checksum		Dummy	

Data: Byte 1 specifies the number of sheets.	0-255 (1 in the example). 0 means line feed only.
Data: Byte 2 indicates the number of line feeds.	Higher-order 4 bits represents the number of feeds before printing. Lower-order 4 bits represents the number after printing. Each value is 00h-0Fh.
* 1 feed = 2.64 mm	
Data: Byte 3 holds the palette values.	Default is 00. Palettes are defined by every 2 bits beginning from high bit. (See Chapter 2, Section 2.3, *Character RAM*.)
Data: Byte 4 is the print density adjustment.	00h-7Fh. Default values are 40h and 80h or greater. 00 < 40h < 7Fh -25% 0% +25%

When printing continuous images from multiple screens, setting the number of line feeds to 0 after one screen's worth of data is printed (9 data packets and a data-end packet) enables printing to be continued from one image to the next without a break.

Cautions Regarding Print Instructions (Caution Required)

- Although applications can print 2-255 pages continuously, this may take a long time. Thus, the user should be provided with a means of halting a print job in progress. (See Section 4.5, *Break Packet*.)

- Whenever possible, the print density data should be backed up to avoid the inconvenience of adjusting the density at each startup.

- If a print instruction packet is set within 100 msec of when the motor is stopped, the position where printing resumes may be incorrect. Always send print instruction packets at least 100 msec after the motor has been stopped.

- Always set the number of line feeds before printing to 1 or greater and the number after printing to 3 or greater, except in the case of the previously mentioned continuous printing , when both values are 0. Other values for these parameters may in result in faulty operation, such as double printing on the same line or failure of the last printed line to reach the paper cutter.

9.4.3 Data Packet

Sends print data that are in character data format. The print data is sent in 1-byte increments for the specified number of bytes.

Example:

88	33	04	00	80	02	Data 0 ~ Data (N-1)	C1	C2	00	00
PA		Header				Data	Checksum		Dummy	

Notification of compression/no compression: (NoError through 0x3FFh):

Maximum number of data bytes is 0x280h (16 (bytes/color) x 20 (colors) x 2 (colors)).

Nine of these packets represent 1 printed sheet. (160 dots x 144 dots)

Byte 2 of the header is the compression/no compression notification byte.

* 1: Compression (* upper 4 bits have no effect)
* 0: No compression

Transmission of compressed data is accomplished by compressing one line at a time— each line consisting of 20 characters horizontally and 2 characters vertically— and sending the number of compressed bytes in order, beginning from the first line.

If the compressed lines exceed 0x280h, the non-compressed data is sent as is (mixture of compressed and non-compressed packets). If the extended data do not fill an entire line when the packets are processed, the printer returns a packet error.

If an instruction to stop printing is received while print data is being sent, an initialization packet can be sent instead of the next data packet.

One Game Boy screen of data is represented by 9 data packets. However, a data-end packet can be sent even if the number of data packets sent is less than 9. In this case, the printer will print only the number of lines received. Line feeds can be performed by sending a data-end packet with no data packet and issuing a print instruction. The printer will then feed the number of lines indicated by the instruction.

Sending the following print instruction packet with a data-end packet but no data packet would specify that 5 sheets be printed, with 1 line feed before printing and 3 line feeds after printing, and that the pre-printing line feeds be ignored. The number of line feeds performed would therefore equal the product of number of sheets to be printed and the number of post-printing line feeds specified. Thus, in this example, the number of line feeds would be 15.

Example:

88	33	02	00	04	00	05	13	E4	40	42	01	00	00

9.4.4 Data-End Packet

Actual Data

88	33	04	00	00	00	04	00	00	00
PA		Header				Checksum		Dummy	

This packet has no data section.

A data length of 0 for the data packet header represents the end of the print data. This must always be sent to end print data transmission.

9.4.5 Break Packet

Used to discontinue printing. The break packet is sent by means of the user's instructions and forcibly stops printing. (Printing is halted after 1 line is printed.)

Actual Data

88	33	08	00	00	00	08	00	00	00
PA		Header				Checksum		Dummy	

This packet has no data section.

9.4.6 NUL Packet

A functionless packet for requesting the current status of the printer. The printer may occasionally be halted unintentionally while printing (e.g., paper jam, low battery), so a NUL packet should always first be sent to check the printer's status.

Actual Data

88	33	0F	00	00	00	0F	00	00	00
PA		Header				Checksum		Dummy	

This packet has no data section.

9.4.7 Packet Error

Except in the case of a checksum error, if a packet of one of these types is sent but does not match the specification described, the printer will return a packet error.

9.4.8 Other Packets

Packets other than the types mentioned above are ignored by the printer.

9.5 Printer Status and Packets

The following table shows the packets that can and cannot be sent from the Game Boy to the printer while the printer is in various states.

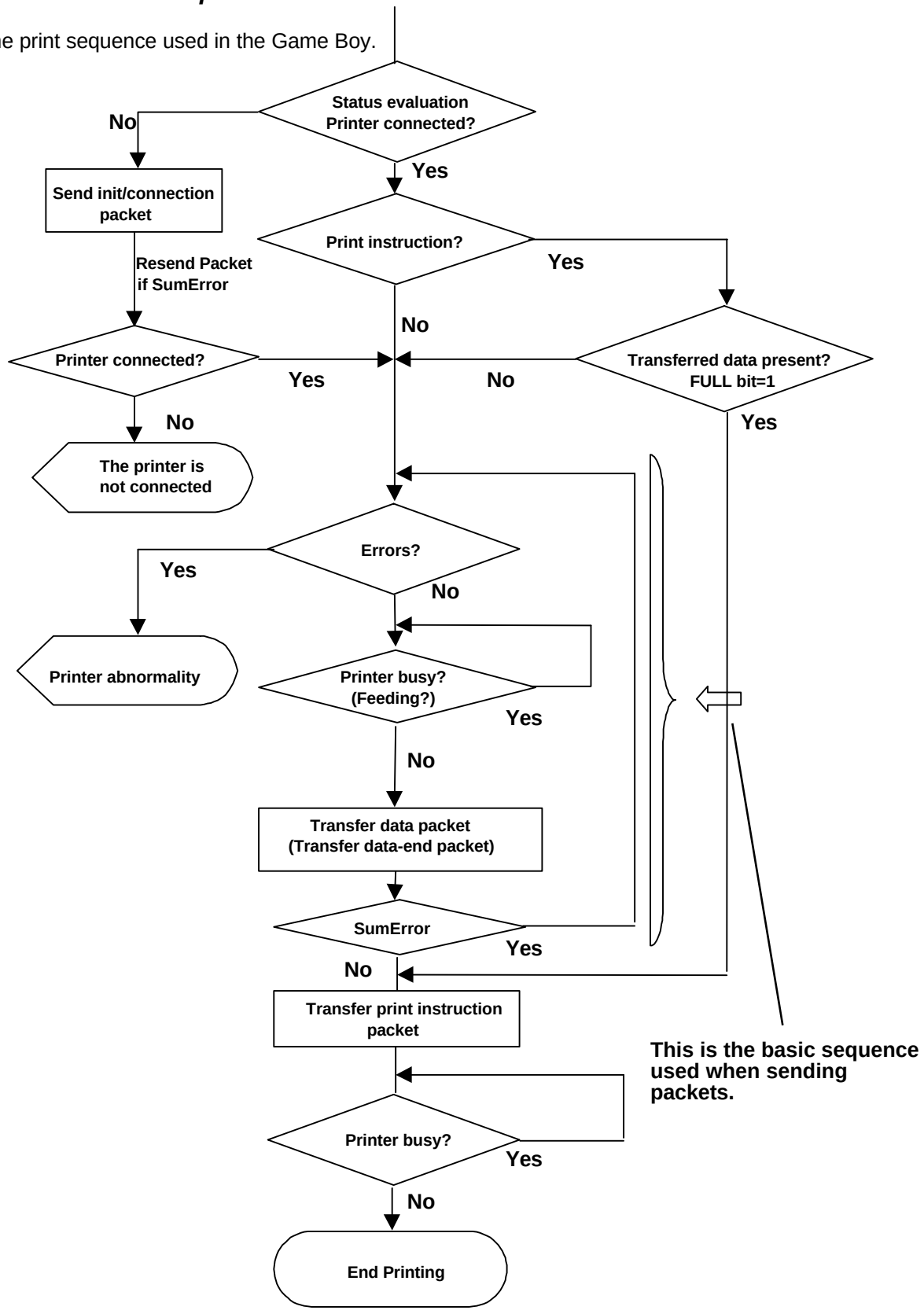
	Disconnected	Immediately after Connected	Print Buffer Full	While Printing	While Feeding
Connection/initialization packet	O	O	O	▲	▲
Print instruction packet	?	x	O	▲	▲
Data packet	?	O	x	▲	▲
Data-end packet	?	O	x	▲	▲
Break packet	?	▲	▲	O	▲
NUL packet	?	O	O	O	O

O = OK; ▲ = ignored; x = packet error; ? = undefined

* The user could push the feed button while data is being transferred. In this case, the entire data packet would be ignored, so the same packet would need to be re-sent.

9.6 Printer Print Sequence

The print sequence used in the Game Boy.



9.7 Processing of Connection Evaluation and Preamble Detection Failure

9.7.1 Connection Evaluation (includes cable disconnection)

To check whether a printer is connected to the Game Boy, it sends a NUL packet. If nothing is connected, the value 0xFF is received; if there is a connection, 0x00 is received.

Game Boy		***Printer***
	Not Connected	
NUL packet sent		No data reception detected
↓		↓
0xFFh received		
↓		No data for 120 ms
Evaluates as not connected		
		↓
		Evaluates as not connected; print data cleared
	Cable connected here	
NUL packet sent		NUL packet detected
↓		↓
0x00h received		Status of 0x00h sent
↓		↓
Evaluates as connected		Evaluates as connected
↓		↓
Connection-check packet sent after 100 msec delay		Connection confirmed, ACK returned
.		.
.		.
.		.
.		.

The printer prepares to print data again; it is not cleared while data is received.

9.7.2 Preamble Detection Failure

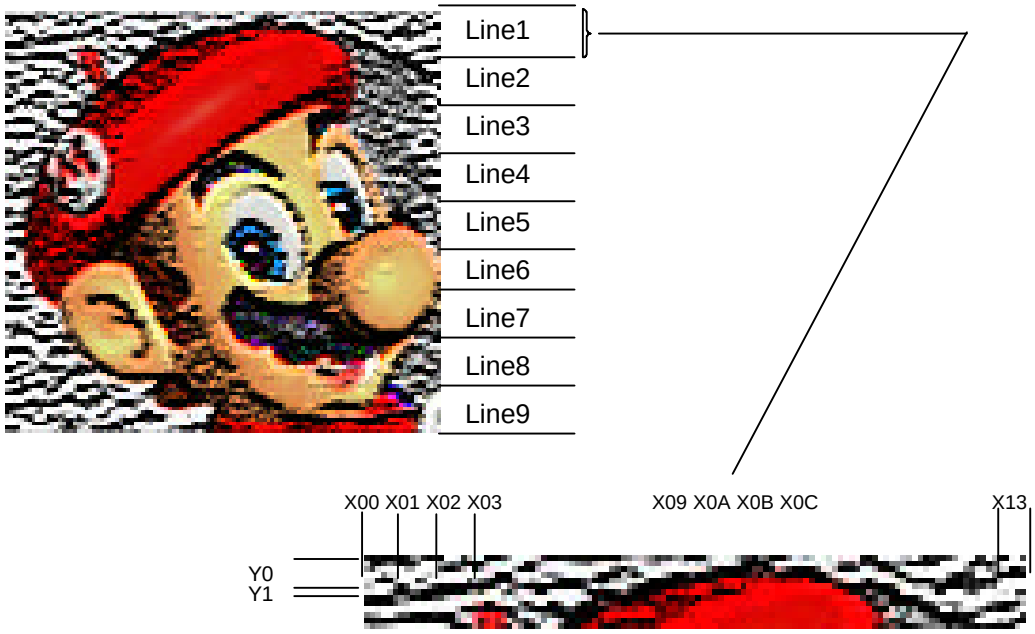
If preamble detection fails during data reception, the flow of the Game Boy and printer sequences are as shown below in parallel.

Game Boy	***Printer***
Printer status normal	Printer normal
↓	↓
Start communication	Wait to receive data
↓	↓
Cable disconnects during data transfer	Data reception
↓	↓
Printer status = 0xFFh	Preamble detection failure
↓	↓
Confirm reset of printer connection	Set status to 0xFFh
↓	↓
Check connection after 100 msec	80 msec delay
	↓
	Printer initialization

9.8 Print Data

The print data transferred in data packets is in character data format.

Printing Example



Transfer Order

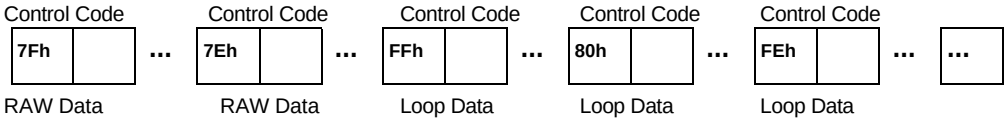
Y0.X00 --> Y0.X01 --> Y0.X02 --> ··· Y0.X13: 2 x 8 x 20 = 140h bytes
Y1.X00 (2 x 8) --> Y1.X01 --> ··· Y1.X13: 2 x 8 x 20 = 140h bytes
Total: 280h bytes

1 CHAR = 2 bytes (higher grayscale, then lower grayscale) x 8 dots vertically

9.9 Compression Algorithm

Compressed data essentially consist of control codes specifying the data type and length and the actual data.

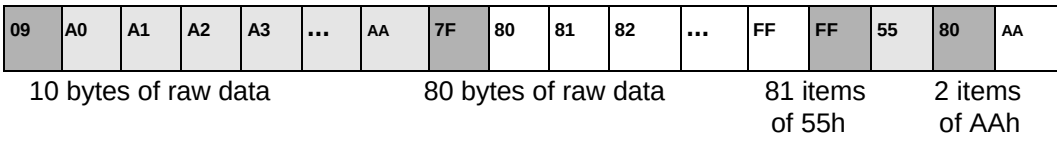
① Control code 1 + raw data
② Control code 2 + loop data



① Control code 1 + Raw data
7Fh: Next 80h bytes are raw data
0h-7Eh (N): Next < N + 1 data items (01h-7Fh) are raw data

② Control code 2 + Loop data
FFh: Repeat the next < 1 byte of data for 81h bytes
80h-FEh: Repeat the next < 1 byte of data for 2 (80) – 80h (FE) items

Example:



9.10 Hardware Specifications

9.10.1 General Specifications

- Printing method: Thermal serial dot
- Print direction: Left to right (facing direction of paper feed)
- Total dot count: 16 x 160 (H x W/line)
- Dot pitch: 0.165 mm x 0.167 mm (H x W)
- Dot dimensions: 0.14 mm x 0.164 mm (H x W)
- Paper feed pitch: 2.64 mm
- Print width: Approximately 26.6 mm
- Printing speed: Approximately 1.1 lines/sec

9.10.2 Dimensions and Weight

- Dimensions: 72.2 mm x 139.5 mm x 56.0 mm (W x D x H)
- Weight: Approximately 190 g (not including battery)

9.11 Miscellaneous

9.11.1 Cautions when Debugging

The printer comes in two types, each made a different manufacturer (Seiko Systems and Hosiden). During final game debugging, the game should be checked with at least 1 printer of each type. The manufacturer can be determined from the serial number on the back of the unit (Printers with PS serial numbers are made by Seiko; those with PH serial numbers are made by Hosiden.) Many of the Seiko printers obtained on the market are the normal Pocket Printer, while many of the printers made by Hosiden are manufactured according to the special Pocket Printer Pikachu Yellow specification. However, depending on the needs of the manufacturers, there is no guarantee that this distinction will hold true in the future. If obtaining a printer proves difficult, please contact Nintendo for a special consultation.

9.11.2 Sample Programs Provided by Nintendo (subroutines)

Modifying a program to suit the intended use is permitted. However, in creating the original program, values for timing and other parameters were calculated to allow normal operation. These parameters must therefore be carefully considered when modifying a program.

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

APPENDIX 1 PROGRAMMING CAUTIONS 273

1.1 Using this Appendix 273

1.2 Programming Cautions Regarding Game Boy 274

1.2.1 LCDC/VRAM 274
1.2.2 Communication 276
1.2.3 Sound 276
1.2.4 Miscellaneous Notes 277

1.3 Programming Cautions Regarding MBCs 280

1.3.1 All MBCs 280
1.3.2 MBC3 280
1.3.3 MBC5 281

1.4 SGB Programming Cautions 283

1.4.1 ROM Data 283
1.4.2 Default Data 283
1.4.3 SOU_TRN Default Data 283

1.5 Programming Cautions Regarding Pocket Printer 284

1.5.1 Transfer Time Intervals 284
1.5.2 Printing Multiple Sheets Continuously 284
1.5.3 Print Density 284
1.5.4 Operation After the Motor is Stopped 284
1.5.5 Feeds 284
1.5.6 Point of Caution During Debugging 284
1.5.7 Sample Program Provided by Nintendo 284

1.6 Miscellaneous 285

1.7 Programming Cautions for U.S. Programmers 285

Revision History

Date	Section	Description
6/21/99	**1.2.2.1** **1.2.2.2**	**Revision in "Communication Rate"** **Revision in "Communication Errors"**
4/25/00	**1.2.1.4** **1.2.1.5** **1.2.1.6** **1.6.1**	**Addition of "Game Boy Work RAM and Stack RAM"** **Addition of "What to do when Backup Data is Destroyed"** **Addition of "Initialization of Backup Memory"** **Addition of "6.1 Blink Standard"**
10/10/00	**1.2.3.3** **1.2.3.4** **1.3.3.4**	**Addition of "Do Not Play Back Audio that uses Sound 1, 2, and 4"** **Addition of "Noise when using Game Boy Advance"** **Change of "Disabling Vibrations for Resets and Pauses" from (Recommended) to (Required)**

APPENDIX 1 PROGRAMMING CAUTIONS

1.1 USING THIS APPENDIX

Purpose and Scope

These programming notes provide information on how to avoid easily made mistakes during program development, information on unique Game Boy programming issues that require special attention, and special issues regarding peripheral devices.

Items Covered in this Manual

Many of the topics covered in this appendix also are covered elsewhere in different chapters of this manual. This appendix consolidates the discussion of these topics. Topics that would be more easily comprehensible to the reader when presented separately will also be discussed in another chapter, even though this may duplicate the discussion in this appendix.

Note: Although these notes were created to make every effort to eliminate potential sources of trouble once on the market, they do not represent a guarantee that various potential problems can be completely avoided.

1.2 PROGRAMMING CAUTIONS REGARDING GAME BOY

Covers:
DMG: DMG, MGB, and MGL
SGB: SGB and SGB2
CGB: CGB

1.2.1 LCDC/VRAM

1.2.1.1 Setting the LCDC to OFF (Recommended)

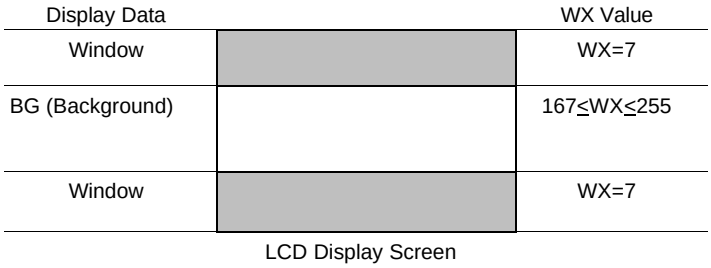
Covers: DMG and CGB
In early DMGs, a black horizontal line appears on the screen if the LCDC is stopped (LCDC register bit 7 ← 0) at any time other than during vertical blanking. Therefore, the LCDC should be set to OFF during V-blanking. If the occurrence of V-blanking cannot be confirmed, the LCDC should be set to OFF when the value of the LY register is 145 (0x91h) or greater. These restrictions do not apply in CGB. Thus, when creating software for use on CGB only, the timing of setting the LCDC to OFF need not be considered.

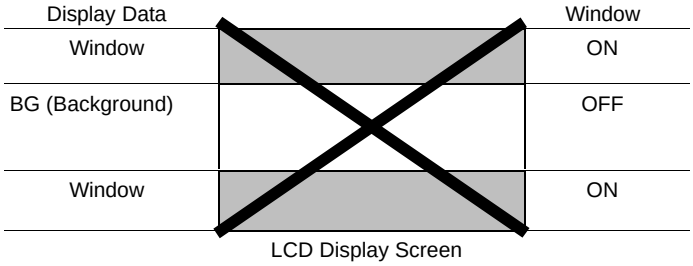
1.2.1.2 Window x-coordinate Register (Required)

Covers: DMG, SGB, and CGB
When the window is displayed, the window x-coordinate register (register WX, address 0xFF4Bh) must be set in the range 7-165. A setting of 0-6 or 166 is prohibited. Specifying a value of 167 or greater causes the window not to be displayed.

1.2.1.3 Displaying Multiple Windows (Required)

Covers: CGB
Multiple windows that divide the screen horizontally into upper and lower areas can be displayed by setting the window x-coordinate register (WX) to a value of 167 or greater during a horizontal blanking period. Attempting to display multiple windows by switching the window ON and OFF during H-blanking may result in the lower window not being displayed.





Reference Notes:

1. Accessing VRAM Outside of a V-blanking period
   In early DMGs, accessing VRAM outside of a V-blanking period would corrupt the screen.

2. Length of H-blanking
   The length of the H-blanking period changes depending on the conditions of OBJ use, so caution is recommended when using H-blanking.

1.2.1.4 Main Work RAM and CPU Work (or Stack) RAM **(Required)**

Covers: DMG, SGB, and CGB
When power is applied to Game Boy, the contents of main Work RAM and CPU Work (or Stack) RAM are undefined. Please verify that your software operates normally, even when the initial value contained in backup memory is different than expected.

The DMG/CGB CPU has an internal monitor program but, when power is applied, control is transferred to the user program as soon as the monitor program is executed. At this time, Work RAM and Stack RAM are not initialized and processing is passed to the user program. Therefore, initial settings of Work RAM and Stack RAM may be different for each Game Boy/Game Boy Color. Values may also change depending upon startup time for a specific Game Boy.

Reference Notes:

Refer to Paragraph 1.3, "Memory Configuration" and 1.4, "Memory Map" for Work RAM and Stack RAM areas.

The initial contents of Work RAM cannot be sorted by manufacturing lot. Therefore, please be careful because they are not included in the test items performed during Lot Check. Basically, the original contents of Work RAM can be maintained because the development system remains energized when it is reset. Hence, you may not notice improper initialization, during development routines.

1.2.1.5 What to do when Backup Data is Destroyed **(Required)**

Covers: DMG, SGB, and CGB
When backup memory is used, verify that the program continues to run as expected when data contained in backup memory is destroyed. It is not possible to completely avoid data destruction if Game Boy is powered off while writing to backup memory.

Reference Notes:

Solutions to Destruction of Backup Data
Error detect ... checksum
Detect abnormal data using a program
Distinguish the destroyed data from new data
Writing complete flag

1.2.1.5 Initialization of Backup Memory **(Required)**

Covers: DMG, SGB, and CGB
Verify that backup memory is initialized normally, even when the contents of backup memory are undefined. Specific values are written to backup memory during manufacturing, but these values cannot be guaranteed.

1.2.2 Communication

1.2.2.1 Communication Rate **(Required)**

Covers: DMG, SGB2, and CGB
Data may be corrupted if the data transfer rate is too high.
The maximum external clock setting should be 512KHz between CGBs.
It should be 500KHz for others including DMG and SGB2.
Also, it should be 256KHz for communication between CGB and DMG.

1.2.2.2 Communication Errors (Recommended)

Covers: DMG, SGB2, and CGB
When using the communication function (infrared), the communicating data may be corrupted by noise and such. Therefore, the program should not go out of control by such data corruption on both the sending and receiving side.
When using the communication function (serial), depending on how the program is made, it is confirmed that communication errors happen rarely.
SIO interrupt processing may be delayed by factors such as the processing of other interrupts. This type of error should be avoided by establishing a proper communication interval that allows a problem-free exchange of data.

1.2.2.3 Effects of Other Infrared Devices (Recommended)

Covers: CGB
Adequate care should be taken to ensure against faulty operation and loss of program control even when infrared communication signal input is received from other game software and devices. Note that such problems may particularly occur in communication between multiple games that use the same subroutines. (Before performing data communication, use means such exchanging a unique key code to check whether the same game is on the other hardware.)

1.2.3 Sound

1.2.3.1 Using Sounds 1, 2 , and 3 (Required)

Covers: CGB
With continuous operation mode selected (bit 6 of NR*4 set to 0) for sounds 1, 2, and 3, if the higher-order frequency data (lower-order 3 bits of NR*4) are changed, the sound length (bits 0-5 of NR*1) must to set to 0 after the frequency data is set. If the sound length is not set to 0, the sound may stop during playback.

1.2.3.2 Using Sound 3 (Required)

Covers: DMG, SGB, and CGB
When sound 3 is used, data should always first be specified for addresses 0xFF30h-0xFF3Fh of waveform RAM. If the initial flag is set during sound 3 operation (sound 3 ON flag = 1), the contents of waveform RAM will be destroyed.

1.2.3.3 Do Not Play Back Audio that uses Sound 1, 2, and 4 (Required)

Covers: DMG, SGB, and CGB
Do not play back audio (speech or song) that uses Sound 1,2, and 4. It cannot be played back normally with AGB's CGB compatible mode.

1.2.3.4 Noise when using Game Boy Advance (Recommended)

Covers: DMG, SGB, and CGB
When you play back audio, by updating the Sound 3 Waveform RAM regularly, noise may occur with AGB's CGB compatible mode. Always confirm it with Game Boy Advance. For example, if Sound 3 is turned ON and OFF (bit 7 of NR30) repeatedly, it could cause noise.

1.2.4 Miscellaneous Notes

1.2.4.1 Using Interrupts (Required)

Covers: DMG, SGB, and CGB
When interrupts are used, the IF register should be cleared before the IE register is set. If the IF register is not first cleared, an interrupt may be generated immediately after interrupts are enabled.

1.2.4.2 Reading Keys (Required)

Covers: DMG
An interval of approximately 18 cycles should be used between output from P14 and P15 and reading of input. Without this interval, normal key input cannot be read.

1.2.4.3 Using the Timer (Required)

Covers: DMG, SGB, and CGB
The timer should be started (TAC start flag set) after the count-up pulse is selected. Starting the timer before or at the same time as the pulse is selected may result in an extra count-up operation at the time of pulse selection.

Example:

```
LD   A,3      ;Selects f/256 as the count-up pulse.
LD   (07),A   ;Sets TAC ← 3
LD   A,7      ;Starts the timer
LD   (07),A
```

If a write instruction is executed for the modulo register TMA with the same timing as the contents of that register are transferred to TIMA as a result of a timer overflow, the same write data also will be transferred to TIMA..

1.2.4.4 Using STOP Mode (Required)

Covers: DMG, SGB, and CGB
When STOP mode is used, all interrupt-enable (IE) flags should be reset before execution of a STOP instruction. Otherwise, if an interrupt is generated during the period of oscillation stabilization (HALT mode) following STOP mode cancellation, HALT mode will immediately be canceled, preventing a stable system clock from being provided.

1.2.4.5 Using Paired Registers (Required)

Covers: DMG, SGB, and CGB
With instructions that use paired registers BC, DE, and HL, such as the following, there is some chance that OAM RAM may be destroyed. Therefore, ensure that these paired registers are not set to a value in the range 0xFE00-0xFE9E.

```
INC   ss          ; ss : BC, DE, HL
DEC   ss
LD    A,(HLI)
LD    A,(HLD)
LD    (HLI),A
LD    (HLD),A
```

1.2.4.6 Using the HALT Instruction (Required)

Covers: DMG, SGB, and CGB
When using a HALT instruction, always add an NOP instruction immediately after the HALT instruction. Not adding the NOP instruction may in rare cases cause the instruction after the HALT instruction not to be executed.

1.2.4.7 Switching the CPU Operating Speed (Recommended)

Covers: CGB
When switching the CPU operating speed, first confirm the current speed by checking the speed flag (bit 7 of register KEY1). In double-speed mode, both the divider (DIV) and timer (TIMA) registers will also be set for double-speed operation.

1.2.4.8 Using Horizontal Blanking DMA (Required)

Horizontal blanking DMA should always be started (bit 7 of HDMA5 set to 1) when the STAT mode is not set to 00. If horizontal blanking DMA is started when STAT mode is 00, depending on the timing, the data in LCD display RAM may be destroyed. In addition, execution of a HALT instruction during horizontal blanking DMA may prevent normal cancellation of the HALT mode or DMA. Therefore, HALT instructions should not be used while horizontal blanking DMA is being started.

1.2.4.9 Using General-Purpose DMA (Required)

General-purpose DMA should be started (bit 7 of HDMA5 set to 0) with the LCDC off or during V-blanking. However, when transferring data during V-blanking, ensure that the transfer period does not overlap with STAT modes 10 or 11.

1.2.4.10 DMA Transfers to OAM (Required)

In DMG and in CGB in DMG mode, when transferring data to OAM by DMA, the user program area (0x00h-0x7FFFh) should not be used as the starting address of the transfer. In some cases, data cannot be transferred from the user program area normally. CGB mode, however, does enable DMA transfers from the user program area.

1.2.4.11 Status Interrupts (Required)

Covers: DMG, SGB, and CGB
When using a status interrupt in DMG or in CGB in DMG mode, register IF should be set to 0 after the value of the STAT register is set. (In DMG, setting the STAT register value changes the value of the IF register, and an interrupt is generated at the same time as interrupts are enabled.)

1.2.4.12 Chattering (Recommended)

Covers: DMG, SGB, and CGB
To prevent buttons from inadvertently being pressed twice, an interval should be provided between key reads. (Although this varies with the software, keys are normally read approximately once per frame.)

1.3 PROGRAMMING CAUTIONS REGARDING MBCS

1.3.1 All MBCs

1.3.1.1 Protecting RAM Data (Recommended)

To protect RAM data, access to RAM should be disabled (RAMG←0x00h) when it is not being accessed.

1.3.2 MBC3

1.3.2.1 Accessing the Clock Counters (Required)

If the clock counters themselves are counted up, accessing of the clock counters by the CPU is performed asynchronously. However, if these operations are performed simultaneously, the clock counters may fail. To prevent this, MBC3 provides an interface circuit for WR signals from the CPR. Use of this circuit necessitates a delay when accessing control register 3 and the clock counter registers (RTC_S, RTC_M, RTC_H, RTC_DL, and RTC_DH). Thus, whenever accessing these registers consecutively, interpose a delay of 4 cycles between accesses.

When reading clock counter data:

- Latch all clock counter data using control register 3.

4-cycle delay required

- Read the data in the clock counter registers.

When writing values to the clock counters:

- Set data in clock counter register RTC_S.

4-cycle delay required

- Set data in clock counter register RTC_M.

4-cycle delay required

- Set data in clock counter register RTC_H.

4-cycle delay required

- Set data in clock counter register RTC_DL.

4-cycle delay required

- Set data in clock counter register RTC_DH.

1.3.2.2 Condensation (Required)

MBC3 uses a crystal oscillator for its clock counter operation, and condensation on the oscillator may halt its oscillation, preventing the clocks from counting up. Once the condensation disappears, the clocks will resume counting up from where they stopped. However, please ensure that the counter stoppage does not result in a loss of program control.

1.3.2.3 Control Register Initialization (Required)

Although control registers 0-3 are initialized (see Section 3.2, Description of Registers) when the Game Boy power is turned on, they are not initialized by a hard reset of Super NES when Super Game Boy is used. Therefore, please be sure to implement a software reset of these registers.

1.3.2.4 Clock Counter Registers (Required)

When commercial GB software that use MBC3 are shipped from the factory, the values of the clock counter registers are undefined. Therefore, please ensure that these registers are initialized.

1.3.3 MBC5

1.3.3.1 Memory Image (Required)

If a memory device is used that uses less than the maximum amount of memory available (ROM: 64 Mbits; RAM: 1 Mbit) , a memory image is generated for the empty bank area. Therefore, please do not develop software that uses an image, because it may cause failures.

1.3.3.2 Specifying External Sound Input (VIN) (Required)

Always use the sound control register (NR50) with bits 7 and 3 (VIN function OFF) set to 0. Because the VIN terminal is used in development of Flash ROMs, using the register with VIN set to ON will produce sound abnormalities.

1.3.3.3 Disabling Vibration Using the SGB, SGB2, or N64 Transfer Pak (Recommended)

When MBC5 with rumble feature is used by SGB, SGB2, or the N64 Transfer Pak, vibration should be turned off by the program to prevent failures caused by a faulty connection. For methods of recognizing SGB and SGB2, see Chapter 6, section 4.2, Recognizing SGB. With the N64 Transfer Pak, vibration is controlled by the N64 software. Therefore, N64 software programs that support MBC5 should not write data to bit 3 of the RAM bank register.

1.3.3.4 Disabling Vibrations for Resets and Pauses (Required)

Vibration should be halted during resets and pauses.
When power is turned on, the hardware should not be vibrated until some input is received from the controller.

1.3.3.5 Limiting the Period of Continuous Vibration (Recommended)

To prevent physical effects in the user such as numbness as a result of continuous vibration, limit the duration of continuous vibration as indicated below, regardless of the vibration strength.

- Limit the duration of continuous vibration to 1 minute.
- If the nature of the game makes longer periods of continuous vibration unavoidable, limit these periods to 3 minutes.

1.3.3.6 Rumble Feature Selection (Recommended)

The user should be allowed to set the rumble feature to ON or OFF or to select strong, mild, or OFF by means such as an initial-settings screen at the start of the game. In addition, the program should allow the user to easily change these settings even during a game if, for example, they are uncomfortable with the vibration. Such changes also should be allowed a pause.

1.3.3.7 Changes in Vibration Level with Battery Use (Recommended)

If the battery that powers the motor (size AAA alkaline battery) wears out, the perceived vibration level will be reduced even if the requested vibration level remains the same. Therefore, rumble operation should be checked both when the battery is new (1.6 V) and when it is at the end of its life (1.1 V).

1.4 SGB PROGRAMMING CAUTIONS

1.4.1 ROM Data (Required)

To use the functions of SGB (system commands), the following values must be stored in ROM at the locations indicated.
0x146h ← 0x03h, 0x14Bh ← 0x33h

1.4.2 Default Data (Required)

When writing programs that use the functions of SGB, use the initialization routine of the game program to send default data (see Chapter 6) to the register file.

1.4.3 SOU_TRN Default Data (Required)

When using the SOU_TRN system command, send the SOU_TRN default data (see Chapter 6) to the register file before SOU_TRN is used.

1.5 PROGRAMMING CAUTIONS REGARDING POCKET PRINTER

1.5.1 Transfer Time Intervals (Required)

Transfer time intervals vary depending on the manufacturer. The timings indicated in Chapter 9 should be used to avoid faulty operation with a printer from a particular manufacturer.

1.5.2 Printing Multiple Sheets Continuously (Recommended)

Between 2 and 255 sheets can be printed continuously by an application. However, because this may take a long time, the user should be given a means of halting a print job in progress.

1.5.3 Print Density (Recommended)

Because it is very inconvenient to adjust the density each time the program is started, the print density data should be backed up whenever possible.

1.5.4 Operation After the Motor is Stopped (Required)

If a print instruction packet is sent within 100 msec of when the motor is stopped, the print starting position may be incorrect. Therefore, print instruction packets should always be sent at least 100 msec after the motor is stopped.

1.5.5 Feeds (Required)

In setting the number of line feeds to be inserted before and after printing (byte 2 of the data portion of the print instruction packet), always specify a value of 1 or greater for the number of feeds before printing and 3 or greater for the number after printing. Otherwise, problems can arise, such as double printing twice on a single line or failure of the last line of print to reach the paper cutter.

1.5.6 Point of Caution During Debugging (Recommended)

There are two types printers, each made by a different manufacturer (Seiko Instruments and Hosiden). As part of final debugging, the program should be checked with at least one printer of each type.

1.5.7 Sample Program Provided by Nintendo (Recommended)

Modifying the program to suit the intended use is permitted. However, in creating the original program, values for timing and other parameters were calculated to allow normal operation. These parameters must therefore be carefully considered when modifying the program.

1.6 MISCELLANEOUS

1.6.1 Blink Standard (Recommended)

Covers: DMG, SGB, and CGB
We recommend that you ensure that the screen does not blink excessively. In rare cases, some people experience cramps or lose consciousness temporarily when experiencing intense light stimulus or blinking, or while watching a TV screen.

1.7 PROGRAMMING CAUTIONS FOR U.S. PROGRAMMERS

If you are unable to verify that the system is a Super Game Boy, and the Accumulator returns the same value if the game is inserted in the Super Game Boy or original Game Boy hardware, follow the instructions below.

If your game is Super Game Boy (SGB) enhanced, then you just need to use the MLT_REQ function. Otherwise, you must use the SGB libraries to verify if the game is in an SGB. (These libraries are located in the CGB files section of Wario World under SGBlib.zip.) You will need to call the SGBCHK function from these libraries right after the Soft Reset label. To use this function, you must set the ROM Registration area for SGB ($146h) to $03, which allows access to the SGB REgisters. (Don't forget to readjust the Complement Check.)

Also, on the Software Submission sheet, make sure you note that the game has a $03 in address $146, but in the remarks section, explain that the game doesn't use any of the SGB features.

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

APPENDIX 2: REGISTER AND INSTRUCTION SET SUMMARIES 289

2.1 Control Register Summary.....289

2.2 Sound Register Summary 294

2.3 CPU Instruction Set Summary 298

THIS PAGE WAS INTENTIONALLY LEFT BLANK.

APPENDIX 2: REGISTER AND INSTRUCTION SET SUMMARIES

2.1 Control Register Summary

Register	Address	D7	D6	D5	D4	D3	D2	D1	D0	Comment
P1 Port P15-P10	FF00			P15	P14	P13	P12	P11	P10	R/W Control of transfer data by P14, P15
SB Serial transfer register	FF01									R/W Transfer data
SC Serial control	FF02	Transfer start 0: No start 1: Start						Clock speed 0: 8 KHz 1: 256 KHz	Shift clock 0: External 1: Internal	R/W In double-speed mode clock speed also doubled
DIV Divider	FF04	$f/2^{16}$ 64Hz	$f/2^{15}$ 128Hz	$f/2^{14}$ 256Hz	$f/2^{13}$ 512Hz	$f/2^{12}$ 1024Hz	$f/2^{11}$ 2048Hz	$f/2^{10}$ 4096Hz	$f/2^{9}$ 8192Hz	R/W With clear normal by LD instruction: f=4194304 Double speed: f=8388608
TIMA Timer	FF05									R/W Timer unit. Operates at double-speed in double-speed mode
TMA Timer modulo	FF06									R/W Preset register for timer
TAC Timer control	FF07						Timer stop 0: Stop 1: Operate	Frequency selection bit 00: $f/2^{10}$ 10: $f/2^{6}$ 01: $f/2^{4}$ 11: $f/2^{8}$		R/W Normal-speed: f=4194304 Double-speed: f=8388608
IF Interrupt request flag	FF0F				Terminals P10-P13 HIGH	End of serial transfer	Timer overflow	LCDC Controller STAT	V-blank	R/W Bit reset valid
IE Interrupt enable flag	FFFF				Terminals P10-P13 LOW	End of serial transfer	Timer overflow	LCDC Controller STAT	V-blank	R/W 0: Disabled 1: Enabled

Register	Address	D7	D6	D5	D4	D3	D2	D1	D0	Comment
IME Interrupt master enable										Reset with DI; set with EI 0: Disable interrupts 1: Enable interrupts
LCDC LCDC Control	FF40	Controller 0: Stop 1: Operate	WIN Area 0: 9800- 1: 9C00-	Window 0: OFF 1: ON	BG Characters 0: 8800- 1: 8000-	BG Area 0: 9800- 1: 9C00-	OBJ Block 0: 8x8 1: 8x16	OBJ Display 0: OFF 1: ON	BG Display 0: OFF 1: ON	R/W Bit 0 fixed to display BG ON in CGB mode only
STAT LCDC status information	FF41		LCDC status interrupt selection flags Agreement flag selection	Mode 10 selection	Mode 01 selection	Mode 00 selection	LYC agreement 0: 1: LYC=LY	Mode 00: RAM access 10: OBJ search 01: V-blank 11: LCD transfer		R/W Bits 3-6 Interrupt 0: Not selected 1: Selected
SCY Scroll Y register	FF42									R/W 00h – FFh
SCX Scroll X register	FF43									R/W 00h – FFh
LY LCDC y-coordinate	FF44									R y-coordinate during display
LYC LY compare register	FF45									R/W Agreement flag set with LYC=LY
DMA DMA Transfer	FF46									W 00h – DFh Transfer starts at the same time as address set

Register	Address	D7	D6	D5	D4	D3	D2	D1	D0	Comment
BGP BG Palette Data	FF47	Palette data for character dot data 11 in DMG mode.		Palette data for character dot data 10 in DMG mode.		Palette data for character dot data 01 in DMG mode.		Palette data for character dot data 00 in DMG mode.		W
OBP0 OBJ palette data 0	FF48	Palette data for character dot data 11 in DMG mode.		Palette data for character dot data 10 in DMG mode.		Palette data for character dot data 01 in DMG mode.		Palette data for character dot data 00 in DMG mode.		W When attribute bit 4 is 0.
OBP1 OBJ palette data 1	FF49	Palette data for character dot data 11 in DMG mode.		Palette data for character dot data 10 in DMG mode.		Palette data for character dot data 01 in DMG mode.		Palette data for character dot data 00 in DMG mode.		W When attribute bit 4 is 1.
WY Window y-coordinate	FF4A									R/W 0 – 143 Top edge when WY=0
WY Window x-coordinate	FF4B									R/W 7 – 165 Left edge when WY=7
KEY1 CPU speed switching	FF4D	Current speed: 0: Normal 1: Double-speed							Enable speed switching	R/W Switch by setting bit 0 to 1 and issuing a STOP instruction
VBK VRAM bank specification	FF4F								Bank 0: Bank 0 1: Bank 1	R/W Bank 0 selected immediately after a reset signal.
HDMA1 Higher-order address of HDMA transfer source	FF51									W 00h – 7Fh (ROM) A0h – DFh (WRAM)
HDMA2 Lower-order address of HDMA transfer source	FF52									W 0Xh – FXh
HDMA3 Higher-order address of HDMA transfer destination	FF53									W 00h – 1Fh

Register	Address	D7	D6	D5	D4	D3	D2	D1	D0	Comment
HDMA4 Lower-order address of HDMA transfer destination	FF54									W 0Xh – FXh
HDMA5 H-blank and general-purpose DMA control	FF55	DMA selection 0: General purpose 1: H-blank	0 ≤ n ≤ 127 Total number of transferred bytes: 16 x (n+1)							W H-blanking stopped by setting bit 7 to 0; General-purpose stopped by resetting
RP Infrared communication port	FF56	Data read-enable flag 00: Disable 11: Enable						Read data 0:LED-ON 1:LED-OFF	Write data 0:LED-OFF 1:LED-ON	R/W
BCPS Color palette BG write specification	FF68	Increment 0: OFF 1: ON		Palette No. 0 - 7			Palette No. 0 - 3		H/L specification 0: L 1: H	R/W Not incremented automatically with a read
BCPD Color palette BG write data	FF69									R/W
OCPS Color palette OBJ write specification	FF6A	Increment 0: OFF 1: ON		Palette No. 0 - 7			Palette No. 0 - 3		H/L specification 0: L 1: H	R/W Not incremented automatically with a read
OCPD Color palette OBJ write data	FF6B									R/W

Register	Address	D7	D6	D5	D4	D3	D2	D1	D0	Comment
SVBK WRAM Bank specification	FF70						Banks specification 0,1: Specifies bank 1 2-7: Specifies banks 2-7			R/W
OBJ0*1 LCD y-coordinate	FE00									R/W 00h – FFh Top edge when Y= 10h
  LCD x-coordinate	FE01									R/W 00h – FFh Left edge when X=08h
  Character code	FE02									R/W 00h – FFh
  Attribute flag	FE03	Display priority 0: OBJ 1: BG	Vertical flip 0: Normal 1: Flip	Horizontal flip 0: Normal 1: Flip	Palette specification for DMG mode	VRAM bank 0: bank 0 1: bank 1	Color Palette No. 0 - 7			R/W
*1 OBJ1 - OBJ39 same as OBJ0										

The dark frame ☐ indicates a flag or register unique to CGB.

2.2 Sound Register Summary

All values shown in the following table apply to normal mode. The values for double-speed mode should be calculated by doubling the system clock frequency.

	Register	Address	D7	D6	D5	D4	D3	D2	D1	D0	Comment
SOUND 1	NR10	FF10		Sweep time: 010:15.6ms 101:39.1ms 000:OFF 011:23.4ms 110:46.9ms 001:7.8ms 100:31.3ms 111:54.7ms			Sweep increase /decrease 0: + f hi 1: - f low	Number of sweep shifts: 0 - 7			R/W f_{128}=128Hz
	NR11 Duty cycle/sound length	FF11	Waveform duty cycle 00: 12.5% 10: 50% 01: 25% 11: 75%	Sound length data t1 : 0 - 63 Sound length = (64-t1) * (1/256) sec							R/W
	NR12 Envelope	FF12	Initial envelope value: 0x00 – 0x0F Mute when 0x00 Maximum when 0x0F				Envelope U/D 0: Attenuate 1: Amplify	Number of envelope steps N = 0 - 7 Length of 1 step = N*(1/64) sec Envelope function stopped when N=0			R/W Initial value of 00 sets to OFF when in DOWN mode
	NR13 Lower-order frequency data	FF13	Lower-order 8 bits of frequency data								W
	NR14 Higher-order frequency data/ other	FF14	Restart when initialize flag set to 1	Length selection 0: Consecutiv 1: NR11				Higher order 3 bits of frequency data With x =11-bit frequency data, f = 4194304 / (4*2^3(2048-x)) Hz			R/W f = 64Hz - 131KHz

	Register	Address	D7	D6	D5	D4	D3	D2	D1	D0	Comment
SOUND 2	NR21 Duty cycle/sound length	FF16	Waveform duty cycle 00: 12.5% 10: 50% 01: 25% 11: 75%		Sound length t1: 0 - 63 Sound length = (64-t1) * (1/256) sec						R/W
	NR22 Envelope	FF17	Initial envelope value 0x00 – 0x0F Mute when 0x00 Maximum when 0x0F				Envelope U/D 0: Attenuate 1: Amplify	Number of envelope steps N = 0-7 Length of 1 step = N*(1/64) sec Envelope function stops when N=0			R/W Initial value of 00 sets to OFF when in DOWN mode
	NR23 Lower-order frequency data	FF18	Lower-order 8 bits of frequency data,								W
	NR24 Higher-order /other frequency data	FF19	Restart when initialize flag set to 1	Length selection 0: Consecutive 1: NR21				Higher order 3 bits of frequency data With x =11-bit frequency data, f = 4194304 / (4*2^3(2048-x)) Hz			R/W f = 64Hz - 131KHz
SOUND 3	NR30 Sound OFF	FF1A	Sound OFF 0: OFF 1: ON								R/W
	NR31 Sound length data	FF1B	Sound length data t1 : 0 - 255 Sound length = (256-t1) * (1/256) sec								R/W
	NR32 Output level	FF1C		Output level 00: Mute 10: 1/2 01: Max 11: 1/4							R/W
	NR33 Lower-order frequency data	FF1D	Lower-order 8 bits of frequency data								W
	NR34 Higher-order frequency data/other	FF1E	Restart when initialize flag set to 1	Length selection 0: Consecutive 1: NR31				Higher-order 3 bits of frequency data With x =11-bit frequency data, f = 4194304 / (4*2^3(2048-x)) Hz			R/W f = 64Hz - 131KHz

	Register	Address	D7	D6	D5	D4	D3	D2	D1	D0	Comment
SOUND 4	NR41 Sound length data	FF20			Sound length data t1 : 0 - 63 Sound length = (64-t1) * (1/256) sec						R/W
	NR42 Envelope	FF21	Initial envelope value 0x00 – 0x0F Mute when 0x00 Max when 0x0F			Envelope U/D 0: Attenuate 1: Amplify	Number of envelope steps N = 0-7 Length of 1 step = N*(1/64) sec Envelope function stops when N=0				R/W Initial value of 00 sets to OFF when in DOWN mode
	NR43 Polynomial counter	FF22	Polynomial counter clock frequency selection 0000: $f_b * 1/2$ 1100: $f_b * 1/2^{13}$ 1110 0001: $f_b * 1/2^2$ 1101: $f_b * 1/2^{14}$ 1111				Step no. selection 0: 15 steps 1: 7 steps	Selection of frequency dividing ratio f_b 000: $f*1/2^3*2$ 110: $f*1/2^3*1/6$ 001: $f*1/2^3*1/1$ to 000: $f*1/2^3*1/7$			W f = 4.194304MHz
	NR44 Initialize/length	FF23	Restart when initialize flag set to 1	Length selection 0: Consecutiv 1: NR41							R/W
CONTROL	NR50 SO1 / SO2 level	FF24	VIN input 0:SO2 OFF 1:SO2 output	SO2 output level control 000 (min) – 111 (max)			VIN input 0:SO1 OFF 1:SO1 output	SO1 output level control 000 (min) – 111 (max)			R/W
	NR51 Distribution to SO1/SO2	FF25	Sound 4 to SO2	Sound 3 to SO2	Sound 2 to SO2	Sound 1 to SO2	Sound 4 to SO1	Sound 3 to SO1	Sound 2 to SO1	Sound 1 to SO1	R/W 0: No output 1: Output
	NR52 Sound-end flag	FF26	All sounds 0: Stop 1: Play				Sound 4 ON flag	Sound 3 ON flag	Sound 2 ON flag	Sound 1 ON flag	R/W

Waveform RAM

Waveform RAM is made up of waveform patterns consisting of 4 bits x 32 steps.

Address	D7	D6	D5	D4	D3	D2	D1	D0
FF30	Step 0				Step 1			
FF31	Step 2				Step 3			
FF32	Step 4				Step 5			
FF3F	Step 30				Step 31			

2.3 CPU Instruction Set Summary

	MNEMONIC	SYMBOLIC OPERATION	FLAGS CY	H	N	Z	CYCL	OP-CODE 76 543 210	COMMENT
8-Bit Transfer/Input-Output Instructions	LD r,r'	r←r'	--	--	--	--	1	01 r r'	
	LD r,n	r←n	--	--	--	--	2	00 r 110	
								◄— n —►	
	LD r,(HL)	r← (HL)	--	--	--	--	2	01 r 110	
	LD (HL),r	(HL) ←r	--	--	--	--	2	01 110 r	
	LD (HL),n	(HL) ←n	--	--	--	--	3	00 110 110	
								◄— n —►	
	LD A,(BC)	A← (BC)	--	--	--	--	2	00 001 010	
	LD A,(DE)	A← (DE)	--	--	--	--	2	00 011 010	
	LD A,(C)	A←(FF00H+C)	--	--	--	--	2	11 110 010	
	LD (C),A	(FF00H+C)←A	--	--	--	--	2	11 100 010	
	LD A,(n)	A←(n)	--	--	--	--	3	11 110 000	
								◄— n —►	
	LD (n),A	(n) ←A	--	--	--	--	3	11 100 000	
								◄— n —►	
	LD A,(nn)	A←(nn)	--	--	--	--	4	11 111 010	
								◄— n —►	
								◄— n —►	
	LD (nn),A	(nn) ←A	--	--	--	--	4	11 101 010	
								◄— n —►	
								◄— n —►	
	LD A,(HLI)	A←(HL) HL←HL+1	--	--	--	--	2	00 101 010	
	LD A,(HLD)	A←(HL) HL←HL-1	--	--	--	--	2	00 111 010	
	LD (BC),A	(BC) ←A	--	--	--	--	2	00 000 010	

Register	r,r'
A	111
B	000
C	001
D	010
E	011
H	100
L	101

	MNEMONIC	SYMBOLIC OPERATION	FLAGS CY	H	N	Z	CYCL	OP-CODE 76 543 210	COMMENT
	LD (DE),A	(DE) ←A	--	--	--	--	2	00 010 010	
	LD (HLI),A	(HL) ←A HL←HL+1	--	--	--	--	2	00 100 010	
	LD (HLD),A	(HL) ←A HL←HL-1	--	--	--	--	2	00 110 010	
16-Bit Transfer Instructions	LD dd,nn	dd←nn	--	--	--	--	3	00 dd0 001	
							L-ADRS	← n →	
							H-ADRS	← n →	
	LD SP,HL	SP←HL	--	--	--	--	2	11 111 001	
	PUSH qq	(SP-1) ←qqH (SP-2) ←qqL SP←SP-2	--	--	--	--	4	11 qq0 101	
	POP qq	qqL←(SP) qqH←(SP+1) SP←SP+2	--	--	--	--	3	11 qq0 001	
	LDHL SP,e	HL←SP+e	*	*	0	0	3	11 111 000	
								← e →	
	LD (nn),SP	(nn) ←SP_L (nn+1) ←SP_H	--	--	--	--	5	00 001 000	e=-128~+127
							L-ADRS	← n →	
							H-ADRS	← n →	

Register Pair	dd
BC	00
DE	01
HL	10
SP	11

Register Pair	qq
BC	00
DE	01
HL	10
AF	11

	MNEMONIC	SYMBOLIC OPERATION	FLAGS CY	H	N	Z	CYCL	OP-CODE 76 543 210	COMMENT
8-Bit Arithmetic and Logical Operation	ADD A,r	A←A+r	*	*	0	*	1	10 000 r	s is any of r,n,(HL) CYCL 1: s is r 2: s is n or (HL)
	ADD A,n	A←A+n	*	*	0	*	2	11 000 110	
								← n →	
	ADD A,(HL)	A←A+(HL)	*	*	0	*	2	10 000 110	
	ADC A,s	A←A+s+CY	*	*	0	*	1,2	-- --- ---	
	SUB s	A←A-s	*	*	1	*	1,2	-- --- ---	
	SBC A,s	A←A-s-CY	*	*	1	*	1,2	-- --- ---	
	AND s	A←AΛs	0	1	0	*	1,2	-- --- ---	
	OR s	AVs	0	0	0	*	1,2	-- --- ---	
	XOR s	A⊕s	0	0	0	*	1,2	-- --- ---	
	CP s	A-s	*	*	1	*	1,2	-- --- ---	
	INC r	r←r+1	--	*	0	*	1	00 r 100	
	INC (HL)	(HL) ←(HL)+1	--	*	0	*	3	00 110 100	
	DEC r	r←r-1	--	*	1	*	1	00 r 101	
	DEC (HL)	(HL) ←(HL)-1	--	*	1	*	3	00 110 101	
16-Bit Arithmetic Operation Instructions	ADD HL,ss	HL←HL+ss	*	*	0	--	2	00 ss1 001	e=-128~+127
	ADD SP,e	SP←SP+e	*	*	0	0	4	11 101 000	
								← e →	
	INC ss	ss←ss+1	--	--	--	--	2	00 ss0 011	
	DEC ss	ss←ss-1	--	--	--	--	2	00 ss1 011	

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

The flag is affected according to the result of the operation.

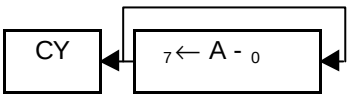
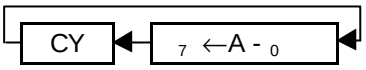
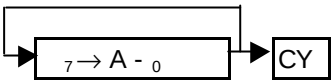
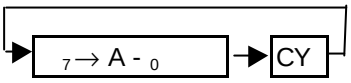
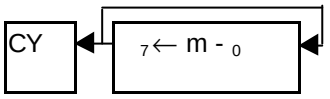
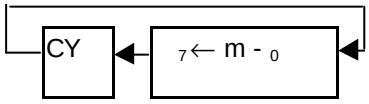
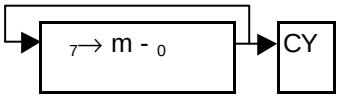
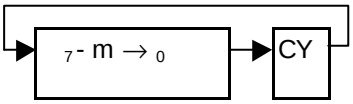
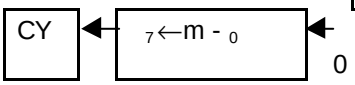
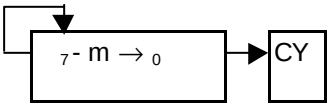
Z: Zero flag. z=1 if the result of the operation is 0

C: Carry/link flag. C=1 if the operation produced a carry from the MSB of the operand or result

H: Half-carry flag.

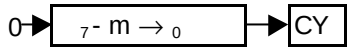
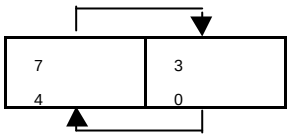
N: Add/Subject flag.

Rotate Shift Instructions

MNEMONIC	SYMBOLIC OPERATION	CY	H	N	Z	CYCL	OP-CODE 76 543 210	COMMENT
RLCA		A_7	0	0	0	1	00 000 111	
RLA		A_7	0	0	0	1	00 010 111	
RRCA		A_0	0	0	0	1	00 001 111	
RRA		A_0	0	0	0	1	00 011 111	
RLC m		m_7	0	0	*	--	-- --- ---	
RL m		m_7	0	0	*	--	-- --- ---	
RRC m		m_0	0	0	*	--	-- --- ---	
RR m		m_0	0	0	*	--	-- --- ---	
SLA m		m_7	0	0	*	--	-- --- ---	
SRA m		m_0	0	0	*	--	-- --- ---	

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

m is any of r, (HL)	CYCL
RLC r	2
RLC (HL)	4
RL r	2
RL (HL)	4
RRC r	2
RRC (HL)	4
RR r	2
RR (HL)	4
SLA r	2
SLA (HL)	4
SRA r	2
SRA (HL)	4

	MNEMONIC	SYMBOLIC OPERATION	FLAGS CY	H	N	Z	CYCL	OP-CODE 76 543 210	COMMENT	
Rotate Shift Instructions	SRL m		m_0	0	0	*	--	-- --- ---	SRL r	2
									SRL (HL)	4
	SWAP m		0	0	0	*	--	-- --- ---	SWAP r	2
									SWAP (HL)	4

	MNEMONIC	SYMBOLIC OPERATION	FLAGS CY	H	N	Z	CYCL	OP-CODE 76 543 210	COMMENT
Bit Operations	BIT b,r	$Z \leftarrow \overline{r_b}$	--	1	0	$\overline{r_b}$	2	11 001 011	
								01 b r	
	BIT b,(HL)	$Z \leftarrow \overline{(HL)}_b$	--	1	0	$(HL)_b$	3	11 001 011	
								01 b 110	
	SET b,r	$r_b \leftarrow 1$	--	- -	--	--	2	11 001 011	
								11 b r	
	SET b,(HL)	$(HL)_b \leftarrow 1$	--	- -	--	--	4	11 001 011	
								11 b 110	
	RES b,r	$r_b \leftarrow 0$	--	- -	--	--	2	11 001 011	
								10 b r	
	RES b,(HL)	$(HL)_b \leftarrow 0$	--	- -	--	--	4	11 001 011	
								10 b 110	
Jump Instructions	JP nn	PC←nn	--	- -	--	--	4	11 000 011	
							L-ADRS	← n →	* No. of cycles is 3 when no cc agreement
							H-ADRS	←n →	
	JP cc,nn	If cc true, PC←nn	--	- -	--	--	4/3	11 0cc 010	
							L-ADRS	← n →	
							H-ADRS	←n →	
	JR e	PC←PC+e	--	- -	--	--	3	00 011 000	
								←e-2 →	
	JR cc,e	If cc true, PC←PC+e	--	- -	--	--	3/2	00 1cc 000	
								←e-2 →	e=-127~+129
	JP (HL)	PC←HL	--	- -	--	--	1	11 101 001	

Bit	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

CC	Condition	Flag
00	NZ	Z=0
01	Z	Z=1
10	NC	CY=0
11	C	CY=1

	MNEMONIC	SYMBOLIC OPERATION	FLAGS CY	H	N	Z	CYCL	OP-CODE 76 543 210	COMMENT
Call/Return Instructions	CALL nn	(SP-1) ←PC_H (SP-2) ←PC_L PC←nn SP←SP-2	--	- -	--	--	6	11 001 101 L-ADRS ◄— n —► H-ADRS ◄— n —►	
	CALL cc,nn	If cc true, (SP-1) ←PC_H (SP-2) ←PC_L PC←nn SP←SP-2	--	- -	--	--	6/3	11 0cc 100 L-ADRS ◄— n —► H-ADRS ◄— n —►	
	RET	PC_L←(SP) PC_H←(SP+1) SP←SP+2	- -	--	--	--	4	11 001 001	
	RETI	PC_L←(SP) PC_H←(SP+1) SP←SP+2	- -	--	--	--	4	11 011 001	
	RET cc	If cc true, PC_L←(SP) PC_H←(SP+1) SP←SP+2	- -	--	--	--	5/2	11 0cc 000	
	RST t	(SP-1) ←PC_H (SP-2) ←PC_L SP←SP-2 PC_H←0 PC_L←P	- -	--	--	--	4	11 t 111	
Gen. Purpose Arithmetic CPU Control Instructions	DAA	Decimal Adjust acc	*	0	--	*	1	00 100 111	
	CPL	A ← $\overline{A}$	- -	1	1	--	1	00 101 111	
	NOP	No operation	- -	--	--	--	1	00 000 000	
	CCF	CY ← $\overline{CY}$	$\overline{CY}$	0	0	--	1	00 111 111	
	SCF	CY←1	1	0	0	--	1	00 110 111	
	DI	IME←0	-	--	--	--	1	11 110 011	
	EI	IME←1	- -	--	--	--	1	11 111 011	
	HALT	Halt		--	--	--	1	01 110 110	
	STOP	Stop	-	--	--	--	1	00 010 000 00 000 000	

Operand	t	(PC_H)	P(PC_L)
0	000	00h	00h
1	001	00h	08h
2	010	00h	10h
3	011	00h	18h
4	100	00h	20h
5	101	00h	28h
6	110	00h	30h
7	111	00h	38h

APPENDIX 3 SOFTWARE SUBMISSION REQUIREMENTS

3.1 The Software Submission Process 307

3.2 Items Required for Submissions 307

3.2.1 Specification Sheet and Check List 307
3.2.2 ROM Data 307
3.2.3 Game Play Videotape/Rating Certificate 307
3.2.4 Screen Text 308
3.2.5 Instruction Manual 308

3.3 Software Verification 308

3.4 Licensee Game Play Videotape Pass/Fail Guidelines 309

3.5 Licensing Screen Information Pass/Fail Guidelines 309

3.6 Common Problems 310

3.7 A Note on Objectionable Material 311

3.8 Instructions for Software Specification Sheet 311

3.9 Character Code List for Game Title Registration 315

3.10 ROM Registration Data Specification 316

3.10.1 Description of ROM Registration Data 317

3.11 Storing Data to the Floppy Disk 321

3.12 Production Software Selection 322

3.13 Development Software Selection 323

3.14 Game Content Guidelines 325

3.15 Game Boy Price Quote Request Form 325

Revision History

Date	Section	Description
4/25/00	3.8	Changed "Programming Features"
	3.12	Changed "Production Software Selection"
	3.13	Changed "Development Software Selection"

APPENDIX 3 SOFTWARE SUBMISSION REQUIREMENTS

3.1 The Software Submission Process

All software submissions to Nintendo of America Inc. must be forwarded to the attention of NOA Product Testing Supervisor. Otherwise, the submission's placement into the testing queue may be delayed. To help reduce a submission's turn-around time, it is suggested that licensees assign a primary contact person for each software submission. All communications with NOA concerning a submission's testing status should be forwarded through this individual. The contact person should also be responsible for notifying any other interested parties.

When a submission is approved, your company's primary contact will be notified immediately in writing.

When a submission is not approved, NOA may send a videotaped copy of the programming problem(s) which prevent(s) the submission from being approved. This is intended to assist the licensee in analyzing the cause of the software problem. It is the licensee's responsibility to send a copy of this tape to any developer(s) of the software. NOA strongly encourages that copies be sent to the software developer(s) as quickly as possible.

Software submissions should be sent to the following address:

Nintendo of America Inc.
Attn: Engineering, Product Testing Supervisor
4820 150th Avenue NE
Redmond, WA 98052
Phone: (425) 861-2674
Fax: (425) 882-3585

3.2 Items Required for Submissions

The following items must be submitted with each Game Boy software submission.

3.2.1 Specification Sheet and SDD/Check List

The appropriate Software Specification Sheet and the Software Destination Declaration/Checklist must be filled out completely and must be correct for the particular program version.

3.2.2 ROM Data

A copy of the ROM data must be submitted in binary format on MS-DOS® 3.5 inch disk(s). The size of the file must be equal to the size of the EP-ROM (i.e., one 4 Meg EP- ROM = one 4 Meg file). Please label each disk and include a description of its contents. (See "Storing Data to the Floppy Disk" below.)

> ***Note:*** ***For software that supports communications, when communications are delayed for more than one hour after the game starts, in addition to the above items, you must submit one set of boards with EP-ROM (or a flash board) in which the game has been advanced to the point where communication can take place.***

3.2.3 Game Play Videotape/Rating Certificate

A video tape containing complete game play is required unless the product has been rated by the Entertainment Software Ratings Board (ESRB). If the product has been rated by the ESRB, then a copy of the rating certificate must accompany the submission and no video tape is needed.

3.2.4 Screen Text

A printed copy of the complete screen text must be submitted.

3.2.5 Instruction Manual

One copy of the instruction manual must be included with your game submission. If, at the time of submission the manual is not complete, (submitted as an intermediate version) then you must submit a list of known bugs.

Note: If any of these items are not satisfied, the program will be rejected and will not be submitted into the approval process until all criteria are met.

3.3 Software Verification

The following verification process will significantly improve the probability of approval of your software.

1. The licensing screen on all submissions should state “LICENSED BY NINTENDO”.
2. Confirm the Licensing Screen information is correct.
3. Check the spelling on the Licensing Screen and Title Screen, as well as the spelling and grammar on the screen text.
4. Confirm the use of a TM (™), circle R (®), or circle C (©) where applicable.
5. Run a “Bypass” Test to assure that when the game is powered up, the Licensing Screen is visible for at least one second, even if any combination of controller buttons are pressed repeatedly. Also “Power-up” the software repeatedly to assure it does so without programming failures.
6. Game characters should be moved in all possible directions or positions, regardless of whether it is required to play the game properly. For instance, if the game does not require going to a particular area to complete the game, go there anyway to assure there are no programming problems in going to that location.
7. The software should be paused many times during the test, as this often causes programming problems to surface.
8. All testing should be recorded onto a videotape, making it easier to review programming problems.
9. The entire attract mode (demo) should be viewed to assure there are no programming problems.
10. Routines designed to assist the programmer or developer in “debugging” the software should be removed from the game prior to submission. This includes routines to determine hardware type.
11. A Game Boy Color dedicated game must include a hardware check upon power-up, which will display the following message when it is connected to a device other than Game Boy Color. The official game title must also be displayed in the upper portion of the display screen.

--<Game Title>--

"This game can only be played on Game Boy Color"

3.4 Licensee Game Play Videotape Pass/Fail Guidelines

1. The licensee game play videotape (if included) must be recorded on a VHS tape, Standard Play speed (SP) for clarity.
2. No editing of the tape is allowed.
3. If more than one tape is needed to show the entire piece of software, then when a second tape begins it must show that the player is in the exact same place as where the first tape left off.
4. No codes or “built-up” characters are allowed.
5. All levels or areas must be completed, in succession.
6. Screen text must have correct grammar and spelling.
7. No deviations from NOA Software Standards Policy may be present.
8. The entire ending credits (if any) must be shown.
9. If the product has been rated by the ESRB, then a copy of the rating certificate must accompany the submission and no videotape is needed.

3.5 Licensing Screen Information Pass/Fail Guidelines

The following Licensing information should be included for all software. This can be displayed on one (1) or two (2) screens.

1. Licensee’s software title.
2. Licensee’s trademark and copyright notice

   (_ 19__ Licensee’s name or copyright owner)
3. LICENSED BY NINTENDO

Example

Tom’s Golf™ or ®

© 1992 ABC Corporation

LICENSED BY NINTENDO

If a blank screen appears for more than two seconds when powered up, Nintendo suggests placing a message or graphic on the screen so that consumers do not think their game is inoperable (e.g., --“Please Wait”--). If a blank screen appears for more than five seconds during game play, a message or graphic should also be placed on the screen.

3.6 Common Problems

Some possible problems that may prevent approval of your software include, but are not limited to the following:

1. Software locks up.
2. Scrambled blocks or characters appear on the screen.
3. The software won't pause.
4. Your character can get stuck somewhere with no possible way to get out.
5. Scrambled graphics at the edges of the screen when the screen scrolls in any direction.
6. Vowels in the passwords or password entry-system.
7. Colored lines at the top or bottom of the screen.
8. Shifting of the screen in any direction.
9. Inconsistent scoring methods.
10. Flashes on screen.
11. Small flickering lines on the screen.
12. Hit or be hit by an enemy but no damage is incurred.
13. Three (3) or four (4) player game can be started without using a four player adapter.
14. Incorrect Licensing Screen; "Licensed by Nintendo" should appear for all formats.
15. Violation of any Programming Cautions in the product programming manual.
16. Communication problems on two-player linkable DMG games.
17. Horizontal or vertical black lines when switching between screens on DMG games.
18. Use of the Nintendo logo or representations of Nintendo products in software without license agreement.
19. The use of the term Super Nintendo or Nintendo when the Super Nintendo Entertainment System or Nintendo Entertainment System is the intended reference, respectively. Use of any term other than Nintendo 64 or N64 when the Nintendo 64 Entertainment System is the intended reference.
20. Character actions are inconsistent (for instance, a character that cannot fly, being able to walk off the edge of a platform and stand in midair).
21. Referring to the Nintendo Control Pad or Control Stick by an unacceptable term, such as; "joypad", "directional control", etc.
22. Referring to the Nintendo Controller by an unacceptable term, such as; "joystick", etc.
23. Referring to the Nintendo Game Pak by an unacceptable term, such as; "Game Cassette", etc.
24. Referring to the Game Boy Game Link by an unacceptable term, such as; "Video Link", etc.

Note: If Licensor approval is required, please assure that this has been finalized before the software submission has been made.

3.7 A Note on Objectionable Material

A copy of the Nintendo “Game Content Guidelines” is included at the end of this document. If you are unsure of whether an item of text or element of a game is within Nintendo Software Standards, you may contact our Engineering Department early in the development process and they will discuss questionable items over the phone. In cases concerning an extensive amount of text, please send it to the attention of NOA Product Testing Supervisor, at the address listed in below, with the questionable items highlighted. The material will be evaluated and you will be contacted within a week to ten days.

Nintendo of America Inc.
Attn: Engineering, Product Testing Supervisor
4820 150th Avenue NE
Redmond, WA 98052
Phone: (425) 861-2674
Fax: (425) 882-3585

3.8 Instructions for Software Specification Sheet

1. Game Title
   Print the planned name for the game. You may use up to 11 characters.
2. Game Code
   Print the product code designated by Nintendo. Use “CGB-P-” for CGB-dedicated software (software that will not operate on a conventional Game Boy). Otherwise, use “DMG-P-”.
3. Language
   Indicate the primary language used for messages, etc. in the game.
4. Communication Mode
   Indicate whether the software has a function which uses an external expansion connector for Game Boy (or Super Game Boy), like a Game Boy communication cable.
5. Software Type
   Indicate whether the game being submitted is DMG exclusive, DMG/CGB compatible, or CGB exclusive.
6. CGB-related Functions
   Check the following items, as appropriate, if you selected "DMG/CGB compatible" or "CGB exclusive" in item 5.

a. Serial Transfer Speed (check all that apply)
   Check all corresponding communication speeds.

b. High Speed ROM Required?
   A high speed ROM is required if CPU double-speed mode (Key 1), horizontal blanking DMA, or general DMA is used.

Note: These 3 functions cannot be used in MBC-1, 2, and 3.

c. IR Communications
   If the software has CGB infrared communications capabilities, please indicate whether the function involves communications with the same game or with a different game. If you select “different game,” include the game title in the parentheses.

7. Overseas Version

   If the game has been, or will be, sold in another country; indicate the product title and product code.

8. Company, Department, Contact Name, Address

   Provide the company name, department, address, phone number, fax number, and the name of a representative that Nintendo should contact with all questions or comments about the product.

9. Submission Date

   Provide the submission date and select the method used for submission.

10. Scheduled Release Date

    Provide the scheduled release date for the game.

11. ROM Registration Data

    Provide the contents registered in the indicated addresses of the master ROM. Refer to “ROM Registration Data Specification” for details. Enter the ASCII code for the characters in areas marked with parenthesis “( )”.

12. Programming Features

    Indicate if special programming is implemented for a specific purpose, such as copy protection. If special programming is implemented, it must be explained in writing.

    a. Initialization of Work RAM and CPU Stack RAM

       The initial value of Work RAM (DMG: 8 Kbytes, CGB: 32 Kbytes) and CPU Stack RAM (127 bytes) is not set at a specific value. Please initialize these areas any time that Game Boy power is turned on or a Reset is initiated. When RAM is installed in a Game Pak, please initialize RAM even if it is used as Work RAM.

       On the specification sheet (or Easy Spec), please check the box next to "Initialized" after confirming that RAM has been initialized. When you choose not to conduct initialization for some specific reason, please check the box next to "Not Initialized" and attach document(s) which explain the situation.

    b. N64 Transfer Pak Support

       If the software is N64 Transfer Pak compatible, indicate the name of the N64 game and its product code. (N64 Transfer Pak is a peripheral device that allows the N64 system to read data from and write to a standard Game Boy Game Pak. This device is not marketed in the U.S. For more information, please contact Nintendo Technical Support.)

    c. Audio Tools

       Please indicate whether or not MusyX Audio Tools were used in the creation of your software. All developers who incorporate MusyX Audio Tools into their game are required to make their best efforts to display a screen credit on the opening licensing screen, which states the following:

       "MusyX Audio Tools Licensed by Factor 5"

13. Game Title Registration

    Enter the game title registered in the master ROM using ASCII characters and their ASCII codes. Also enter the Game Code assigned by Nintendo. Refer to “Character Code List for Game Title Registration” for these entries.

14. Memory Controller

    Indicate the type of memory controller used for this game. If no Memory Controller is used, mark None.

15. Memory Configuration

    Indicate the memory configuration of the game, as follows.

    - ROM: Indicate the ROM size.
    - RAM: Indicate whether or not work RAM is installed in the Game Pak. If work RAM is installed, indicate whether it is used as an expansion device or contained inside an MBC. If it is used as an expansion device, indicate the size of the RAM in the location provided. Also indicate if work RAM requires data back-up (battery). When the MBC-3 Clock Counter function is used, check "Yes" for "Data Back-up", regardless of which box is checked for "RAM".

16. ROM Version

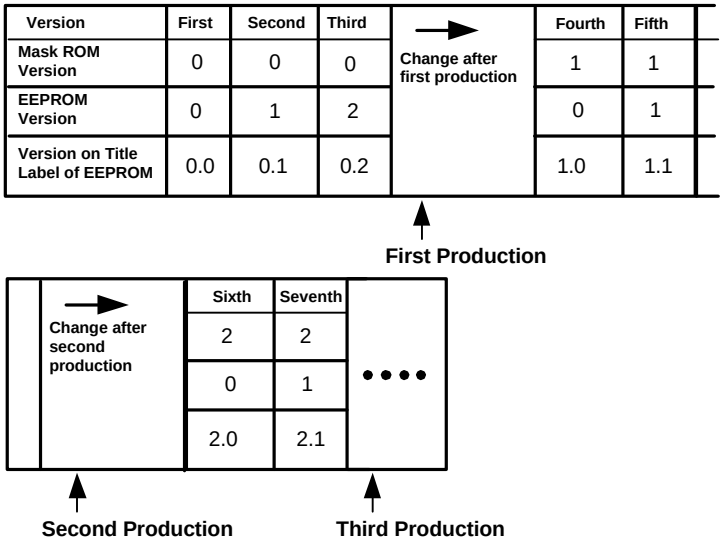
    Mask ROM Version

    - Indicate “0” if submitting the first version of the game.
    - Indicate the next higher number for each revised version after starting production.

    Submission ROM

    - Indicate “0” for the first submission
    - Indicate the next higher number each time the game contents change without updating the Mask ROM version.

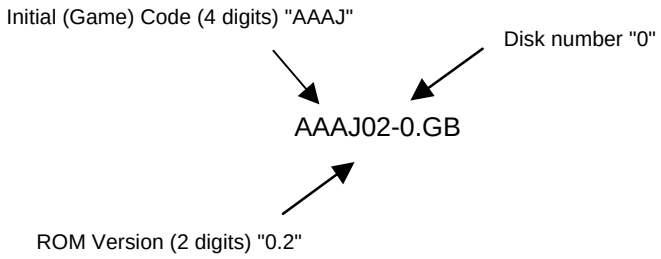
Example



17. File Name and Check Sums

    Print the file name on each disk using the following format: *** **-*.GB

    Example:



> ***Note: The first disk will be numbered "0."***

    If the Game Code is 3 digits (prior to 1994), include an underscore ("_") after the Initial code to bring it to 4 digits. The file name would appear as follows: "AAJ_10-0.GB"

    Enter the check sum of each ROM submitted. To calculate the check sum, add each byte in the ROM data. The lower 2 bytes of the resulting value is the check sum. Enter the check sum for each ROM submitted for the master program and the total of their individual check sums. The total is calculated by adding the individual check sums. This method of calculation is different from the check sum on the ROM Registration Specification.

18. SGB Support

    If the software is designed to use Super Game Boy (SGB) functions, check "Yes." If the software is not specifically designed to use Super Game Boy functions, but will run on SGB, indicate 'No."

    If you checked "Yes" for SGB Support, the SGB Function Code (address 0146H) should contain "03H". If you checked "No", the data contained in address 0146H should read "00H".

    Also, if you checked "Yes" for CGB Support, complete the following 3 items. Do not make any marks in these boxes if you checked "No".

    a. SGB Support Marking

       Check "Yes", if the SGB compatability marking needs to be displayed on product packaging. Otherwise, check "No".

    b. SGB Competition Mode

       Indicate whether the game contains a multi-player function for SGB, by checking the appropriate box.

    c. Program Transfer to Super NES

       Indicate whether or not the program is transferred to the S-CPU for execution as a unique program on the Super NES.

3.9 Character Code List for Game Title Registration

	00	10	20	30	40	50	60	70	80	~	F0
0			SP	0	@	P					
1			!	1	A	Q					
2			"	2	B	R					
3			#	3	C	S					
4			$	4	D	T					
5			%	5	E	U					
6			&	6	F	V					
7			'	7	G	W					
8			(	8	H	X					
9			)	9	I	Y					
A			*	:	J	Z					
B			+	;	K	[					
C			,	<	L	¥					
D			-	=	M	]					
E			.	>	N	^					
F			/	?	O	_					

Note 1: Do not use characters in shaded areas.

Note 2: "SP" means space.

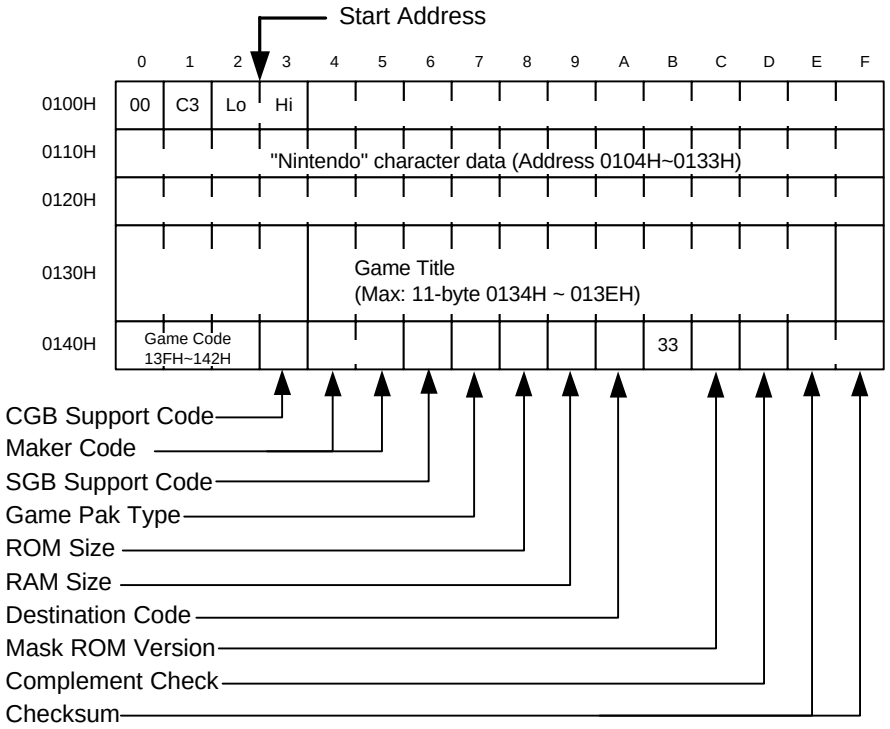
Example: If ASCII character is A, ASCII code is 41.

3.10 ROM Registration Data Specification

Enter information regarding the game title and Game Boy software specifications at the indicated addresses in ROM.

The ROM registration data address is 80 bytes of CPU memory (0100H ~ 014FH).

ROM registration data is stored using the following format.



Note: The following data will be stored in Game Boy Memory for* all *Game Boy software.

0100H = 00H
0101H = C3H
014BH = 33H
0104H~0133H = “Nintendo” character data

3.10.1 Description of ROM Registration Data

1. Start Address (0102H, 0103H)

   The Game Boy (Super Game Boy) program starts after Initial Program Load (IPL) is run on the CPU. The low byte of the starting address is stored first, then the high byte.

2. “Nintendo” Character Data (0104H~0133H)

   Register the character pattern of “Nintendo” to be displayed when the Game Boy is turned on. The following hexadecimal data must be store since IPL verifies it when the program begins.

3. Game Title (0134H~013EH)

   Store the game title (up to 11 characters) using ASCII code. The table “Character Code List for Game Title Registration” is provided for your convenience. Use code 20H for a space and code 00H for all unused areas in the game title. Please use only those characters listed in the provided table when registering a game title. The game title registered should be close to the title under which the game will be marketed. Please do not register a tentative name which is used for development.

4. Game Code (013FH~0142H)

   Store the 4 character game code, assigned by Nintendo, using ASCII code from the table used in item 3. Please use only "upper case" letters, listed in the provided table, when registering a game code.

   Example:

   When the Game Code is "APCJ", the following codes would be stored.

   41H('A') → Address 013FH

   50H('P') → Address 0140H

   43H('C') → Address 0141H

   4AH('J') → Address 0142H

   This requirement only applies to new titles. If the program is changed and a master ROM resubmitted for a game title which has already been marketed, it is not necessary to insert a game code for this submission. (If the Game Code is added to an existing game, please be aware of potential problems with software verification routines in serial communication protocols or N64 Transfer Pak routines. For example, the Game Titles for the old version and the new version MAY be different, causing the new version to be unrecognized by the software verification routine.)

5. CGB Support Code (0143H)

   Store the code which distinguishes between games that are CGB (Game Boy Color) compatible, and those that are not.

Address 143H	Denotation
00H	CGB Incompatible
80H	CGB Compatible
C0H	CGB Exclusive

CGB Incompatible: A program which does not use CGB functions, but operates with both CGB and DMG (Monochrome).

CGB Compatible: A program which uses CGB functions, and operates with both CGB and DMG.

CGB Exclusive: A program which uses CGB functions, but will only operate on a Game Boy Color unit (not on DMG/MGB). If a user attempts to play this software on Game Boy, a screen must be displayed telling the user that the game must be played on Game Boy Color.

6. Maker Code (0144H, 0145H)

   Enter the 2-digit ASCII code assigned by Nintendo. Contact Product Testing, if in doubt. All letters must be in upper case. For example;

   If Maker Code is 01, the ASCII code for 0 (30H) is stored at 0144H and the ASCII code for 1 (31H) is stored at 0145H.

   If Maker Code is FF, the ASCII code for F (46H) is stored at 0144H and 0145H.

7. SGB Support Code (0146H)

   Store the Function Code for the game program. Use the table below.

0146H	Super Game Boy Function
00H	Game Boy (will also run on Super Game Boy)
03H	Uses Super Game Boy Functions

Note: In order to use Super Game Boy functions, the following data must be registered.

0146H = 03H and 014BH = 33H

8. Software Type (0147H)

   Store the appropriate code for the type of software (Game Pak parts configuration) being used.

Address 0147H	**Parts Configuration**								
	ROM	**MBC-1**	**MBC-2**	**MBC-3**		**MBC5**		**SRAM**	**Backup Battery**
				W/ RTC	No RTC	No Rumble	W/ Rumble		
00H	X								
01H	X	X							
02H	X	X						X	
03H	X	X						X	X
04H									
05H	X		X						
06H	X		X						X
07H									
08H	X							X	
09H	X							X	X
0FH	X			X					X
10H	X			X				X	X
11H	X				X				
12H	X				X			X	
13H	X				X			X	X
19H	X					X			
1AH	X					X		X	
1BH	X					X		X	X
19H	X						X		
1AH	X						X	X	
1BH	X						X	X	X

9. ROM Size (0148H)

   Store the code for the program ROM size from the table below.

0148H	ROM Size
00H	256 KBit
01H	512 KBit
02H	1 MBit
03H	2 MBit
04H	4 MBit
05H	8 MBit
06H	16 MBit
07H	32 Mbit
08H	64 Mbit

10. External RAM Size (0149H)

    Store the code for the size of external RAM installed in the Game Pak.

Address 149	RAM Size
00H	No RAM or MBC2
01H	-------
02H	64 KBit
03H	256 KBit
04H	1 Mbit

11. Destination Code (014AH)

    Store the code from the table below which indicates where the product will be marketed.

Address 147	Destination
00H	Japan
01H	All Others

12. Mask ROM Version N0. (014CH)

    The mask ROM version number starts from 00 and increases by 1 for each revised version sent after starting production.

13. Complement Check (014DH)

    After all the registration data has been entered (0134H~014CH), add 19H to the sum of the data stored at addresses 0134H through 014CH and store the complement value of the resulting sum.

    (0134H) + (0135H) +...+ (014CH) + 19H + (014DH) = 00H

14. Check Sum Hi and Lo

    The check sum, excluding the value of 014EH and 014FH, is stored here.

    Check sum Hi and Lo will be different from the Total Check Sum.

    014EH = Upper

    014FH = Lower

3.11 Storing Data to the Floppy Disk

1. Use MS-DOS® 3.5 inch, 2HD disk(s).
2. The data must be submitted in binary (ROM) format. Do not compress the data. The maximum amount of data stored on each floppy should be 8Mbit.
3. The file name should be formatted as described in item #17 of "Instructions for Game Boy Software Specification Sheet - File Name and Check Sums."
4. Place a label describing the content of each disk as shown below.

Company name: Nintendo Co., Ltd.

Product name: Mario's Pikurosu

Product code: DMG-P-APCJ (JPN)

File name: APCJ00-0.GB

Check sum: ABCD

Date: 1998/8/1

3.12 Production Software Selection

MBC \ ROM SIZE / SRAM SIZE		256K	512K	1M	2M	4M	8M	16M	32M	64M	Comments
None	None	○									
	64K	▲									With or without backup battery
MBC-1	None		○	○	○	○	○*1	▲*1			
	64K		○	○	○	○	○*1	○*1			With or without backup battery
	256K	▲	○	○	○	○					With or without backup battery
MBC-2	None	▲	○	○	○						With backup battery only
MBC-3 W/RTC	None	▲	▲	▲	▲	▲	▲	▲			With backup battery only
	64K	▲	○	○	○	○	○	▲			With backup battery only
	256K	▲	▲	▲	▲	▲	○	○			With backup battery only
MBC-5	None		(▲)*2	(▲)*2	○	○	○	○	○	(○)	
	64K		(▲)*2	(▲)*2	○	○	○	○	○	(○)	With or without backup battery
	256K	(▲)*2	(▲)*2	(▲)*2	○	○	○	○	○	(○)	With or without backup battery
	1M	(▲)*2	(▲)*2	(▲)*2	▲	▲	▲	○	○	(○)	With or without backup battery
MBC-5/ Rumble	None	(▲)*2	(▲)*2	(▲)*2	▲	○	○	○	○	(○)	
	64K	(▲)*2	(▲)*2	(▲)*2	▲	○	○	○	○	(○)	With or without backup battery
	256K	(▲)*2	(▲)*2	(▲)*2	▲	○	○	▲	▲	(▲)	With or without backup battery

○ : Board Available
If a price quote is necessary, please submit a "Game Boy Price Quote Request Form" to NOA Licensing Dept.

▲ : Board Not Available
If required, please submit a "Game Boy Price Quote Request Form" to NOA Licensing Dept., approximately 5 months before scheduled software submission.

( ) : At the present time, a mask ROM cannot be prepared. If necessary, please contact NOA Licensing Dept. at (425)861-2091.

[Notes] MBC-1, 2, and 3 do not support Game Boy Color double-speed mode (including H-DMA and General Purpose DMA. Please refer to your Programming Manual.
*1 There are some restrictions in memory mapping when MBC-1 ROM Size is 8M or larger. Please refer to "Memory Controllers" in your Programming Manual.
*2 For MBC-5 with ROM of 1M or less, a mask ROM supporting CGB double-speed mode can not be prepared. Double-speed mode is supported by ROM of 2M or larger.

3.13 Development Software Selection

MBC \ ROM SIZE / SRAM SIZE		256K	512K	1M	2M	4M	8M	16M	32M	64M	Comments
None	None	1									
MBC-1	None		2	3							
	64K/None	5				6					• Built-in 64K SRAM With or without backup battery
				4							
	256K/64K/None			7							• Built-in 256K SRAM With or without backup battery
MBC-2	None			8							
MBC-3	256K/64K/None			9							• RTC Function • Built-in 256K SRAM With or without backup battery
							10				
				11							• Built-in 16M Flash ROM • Built-in 256K SRAM With or without backup battery
MBC-5	1M/256K/64K/None			12							• Built-in 32M Flash ROM • Built-in 1M SRAM With or without backup battery
	256K/64K/None			13							• Built-in 32M Flash ROM • Rumble Function • Built-in 256K SRAM With or without backup battery
	1m/256K/64K/None			14							• Built-in 64M Flash ROM • Rumble Function • Built-in 1M SRAM With or without backup battery

	Product Names (*1)		Memory Specifications (*2)	Comments
	Board Name	**Product Code**		
1	DMG-256K-EPROM	E200225	EPROM : 27C256	EPROM not included
2	MBC1-512K-EPROM	E200241	EPROM : 27C512	
3	MBC1-1M to 2M-EPROM	E200233	EPROM : 27C101/27C2001 (Can use 301 type) (*3)	
4	MBC1-1M to 2M-EPROM+64K	E200530	EPROM : 27C101/27C2001/27C4001	
5	MBC1-Multichecker	E200191	EPROM : 27C256/27C512/27C101/27C301	
6	MBC1-4M to 16M-EPROM+64K	E200654	EPROM : 27C4001	
7	MBC1-1M to 4M-EPROM+256K	E200605	EPROM : 27C101/27C2001/27C4001	
8	MBC2-1M to 2M-EPROM	E200258	EPROM : 27C101/27C2001 (Can use 301 type) (*3)	
9	MBC3-4M-ROM2-256K	E201025	EPROM : 27C101/27C2001/27C4001/27C801	
10	MBC3-8M- ROM2-256K	E201371	EPROM : 27C8001	
11	DMG-MBC3-16M-FLASH	E201561	Built-in 16M Flash Memory +256K RAM Backup Memory	Requires DMG Flash ROM Gang Writer or CGB Emulator
12	DMG-MBC5-32M-FLASH	E201264	Built-in 32M Flash Memory +1M RAM Backup Memory	
13	DMG-MBC5-32M-R-FLASH	E201272	Built-in 32M Flash Memory (with Rumble Pak) +256K RAM Backup Memory	
14	DMG-MBC5-64M-FLASH	E201363	Built-in 64M Flash Memory +1M RAM Backup Memory	

[Notes] MBC-1, 2, and 3 do not support Game Boy Color double-speed mode (including H-DMA and General Purpose DMA. Please refer to your Programming Manual.
There are some restrictions in memory mapping when MBC-1 ROM size is 1M or larger. Please refer to "Memory Controllers" in your Programming Manual.

*1 : When ordering, please indicate both the board name and product code to NOA Licensing Dept. at (425)861-2091.

*2 : For the EPROM specification, please use the described specification, above, or something with the same pin configuration.

*3 : Can support both types for land switching on the board.

3.14 Game Content Guidelines

The following Game Content Guidelines are presented for assistance in the development of authorized game paks (i.e., both Nintendo and licensee game paks) by defining the types of themes inconsistent with Nintendo's corporate philosophy. Exceptions may be made when an objectional item is necessary to maintain the integrity of the product or the games' theme. Nintendo will only approve products (i.e., audio-visual work, packaging and instruction manuals) which do not:

- contain sexually explicit content including but not limited to nudity, rape, sexual intercourse and sexual touching; for instance, Nintendo does not allow bare-breasted women in its games, however, mild displays of affection such as kissing or hugging are acceptable.
- contain language or depictions which specifically denigrate members of any race, gender, ethnicity, religion or political group.
- depict gratuitous or excessive blood or violence. Nintendo does not permit depictions of animal cruelty or torture.
- depict verbal or physical spousal or child abuse.
- permit racial, gender, ethnic, religious or political stereotypes; for example religious symbols such as crosses will be acceptable when fitting into the theme of the game and not promoting a specific religious denomination.
- use profanity, obscenity or incorporate language or gestures that are offensive by prevailing public standards and tastes.
- promote the use of illegal drugs, smoking materials, tobacco and/or alcohol; for example Nintendo does not allow an unnecessary beer or cigarette advertisement anywhere in a product, however Sherlock Holmes smoking a pipe would be acceptable as it fits the theme of the game.

3.15 Game Boy Price Quote Request Form

A Game Boy Price Quote Request Form is included on your Game Boy Software Development Kit (SDK) CD in PDF format. Please FAX this form to Nintendo of America Inc., Attn.: Juana Tingdale, Licensing Department, (206) 861-2173, when requesting a Software Type which is not listed in the previous tables.

THIS PAGE WAS INTENTIONALLY LEFT BLANK